Beweis

- Sei r die optimale Schlußzeit für die k längsten Aufgaben.
- Wir können $F(I) > r$ annehmen, denn sonst $F(I) = F^*(I)$.
- Sei wieder o.B.d.A. $t_1 \geq \cdots \geq t_n$.
- Wir nehmen $n > k$ an (sonst $F(I) = F^*(I)$).
- Wir nehmen $n > m$ an (sonst $F(I) = F^*(I)$).

Wir wählen ein $j > k$, so daß die Bearbeitung der Aufgabe j zum Zeitpunkt $F(I)$ endet.

Falls es so ein j nicht gibt, dann gilt schon $F(I) = F^*(I)$.

Gibt es aber so ein j, dann müssen alle Prozessoren bis zum Zeitpunkt $F(I) - t_j$ beschäftigt sein.

Wegen $t_{k+1} \geq t_j$ sind insbesondere alle Prozessoren auch bis zum (früheren) Zeitpunkt $F(I) - t_{k+1}$ beschäftigt.

Daher gilt

$$\sum_{i=1}^{n} t_i \geq m(F(I) - t_{k+1}) + t_{k+1},$$

denn alle Prozessoren sind je $F(I) - t_{k+1}$ Schritte beschäftigt und einer muß noch Aufgabe t_{k+1} danach erledigen.

Beweis (Fortsetzung)

Aus

$$\sum_{i=1}^{n} t_i \geq m(F(I) - t_{k+1}) + t_{k+1},$$

folgt

$$F^*(I) \geq \frac{1}{m} \sum_{i=1}^{n} t_i \geq F(I) - \frac{m-1}{m} t_{k+1}$$

und schließlich

$$F(I) - F^*(I) \leq \frac{m-1}{m} t_{k+1}. \qquad (*)$$

Um $\dfrac{F(I) - F^*(I)}{F^*(I)}$ abschätzen zu können, brauchen wir noch eine (gute) untere Schranke für $F^*(I)$.

Beweis (Fortsetzung)

Wir betrachten die ersten $k+1$ Aufgaben. Es gibt nur m Prozessoren.

Also muß es einen Prozessor geben, der mindestens $\lceil (k+1)/m \rceil = 1 + \lfloor k/m \rfloor$ von diesen $k+1$ Aufgaben allein erledigt.

Wegen $t_1 \geq \ldots \geq t_n$ dauert es mindesten t_{k+1} Schritte, jede dieser Aufgaben zu erledigen.

Es gibt also einen Prozessor, der $(1 + \lfloor k/m \rfloor)t_{k+1}$ Schritte lang beschäftigt ist.

$$F^*(I) \geq (1 + \lfloor k/m \rfloor)t_{k+1} \qquad (**)$$

Beweis (Fortsetzung)

$$F(I) - F^*(I) \leq \frac{m-1}{m} t_{k+1}. \qquad (*)$$

$$F^*(I) \geq (1 + \lfloor k/m \rfloor) t_{k+1} \qquad (**)$$

Kombinieren wir $(*)$ und $(**)$ erhalten wir

$$\frac{F(I) - F^*(I)}{F^*(I)} \geq \frac{(m-1)/m}{1 + \lfloor k/m \rfloor} = \frac{1 - 1/m}{1 + \lfloor k/m \rfloor}.$$

□

Ein PTAS für Scheduling

Um ein PTAS zu erhalten, müssen wir bedenken, daß ϵ Teil der Eingabe ist.

Daraus muß der Algorithmus ein geeignetes k berechnen, mit

$$\frac{1-1/m}{1+\lfloor k/m \rfloor} \leq \epsilon.$$

Das funktioniert beispielsweise, wenn wir ein k wählen, daß grösser als $(m-1)/\epsilon - m$ ist.

Wie groß ist die Laufzeit? Sie ist $O(n \log n + m^k)$, wenn wir den exakten Teil mit einem Branch-and-Bound-Algorithmus lösen.

Das entspricht $O(n \log n + m^{(m-1)/\epsilon - m})$.

Für jedes feste m haben wir ein PTAS.

Frage: Was passiert wenn m nicht fest ist?