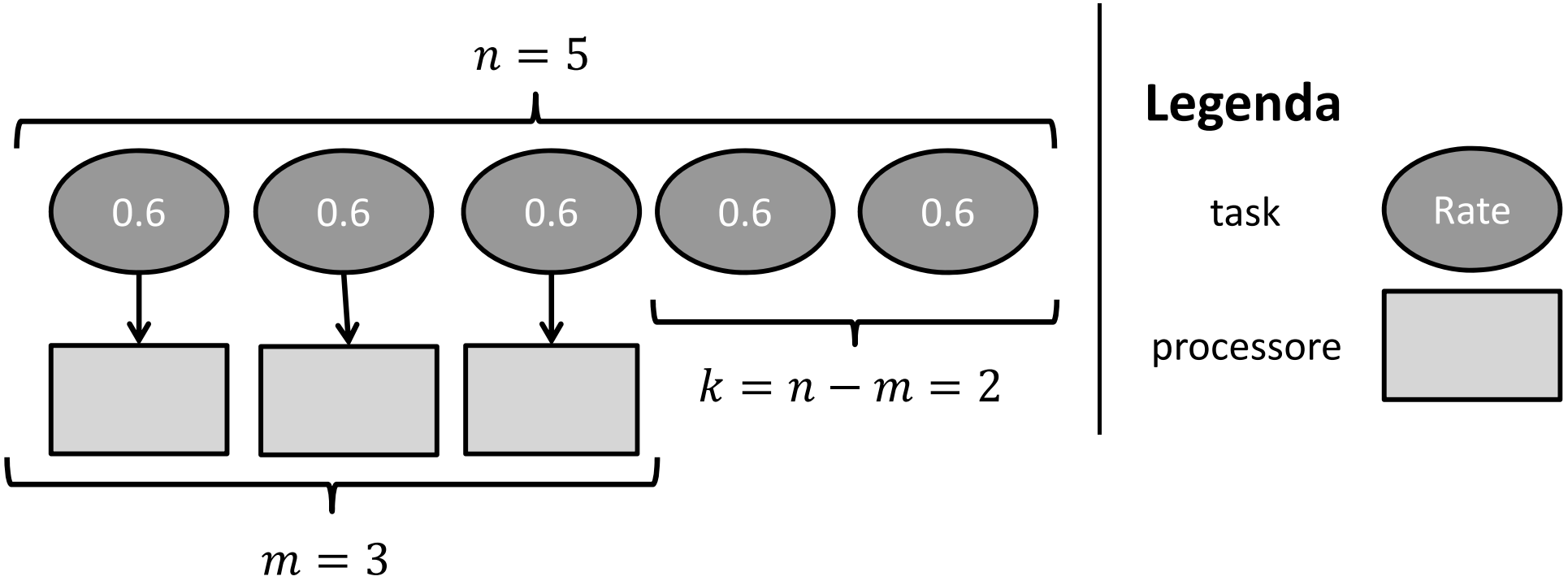
Assunzioni RUN

Parametri di modello

- m processori
- *Implicit-deadline task* $\tau_i, i \in \{1..n\}$ con $n = m + k, k \geq 0$
- *Fully utilized system*: non ci devono essere tempi morti
- Ciascun task ha *fixed rate* $R_i = \frac{(d_i - r_i)}{C_i}$ dove r_i è il *release*
- Task indipendenti
- Processori omogenei
- *Migration* e *preemption* assunti avere zero costo aggiuntivo
- $\sum_{i=1}^{n} R_i \leq m$

  $\Rightarrow$ SI PUO' FARE! Ma come? Con quale Σ valido?

Esempio (1/7)



- $\sum_{i=1}^{n=5}(R_i) = 3 \Rightarrow$ servono almeno 3 processori
- Cerchiamo lo schedule Σ del **sistema S** $\{\{\boldsymbol{\tau_i}\}, \boldsymbol{m}\}$

Sistema duale

L'algoritmo RUN crea l'**insieme duale** dei *task*

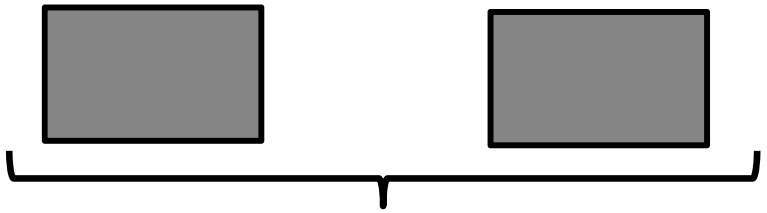
- per ogni *task* τ_i, τ_i^* esegue quando τ_i è *idle* e viceversa
- due *schedule* Σ e Σ^* sono duali quando

$$\forall \mathrm{t}, \tau_i \in \Sigma(t) \text{ se e solo se } \tau_i^* \notin \Sigma^*(t)$$

Tale insieme di *task* viene eseguito nell'**architettura virtuale** del sistema originale (sistema duale)

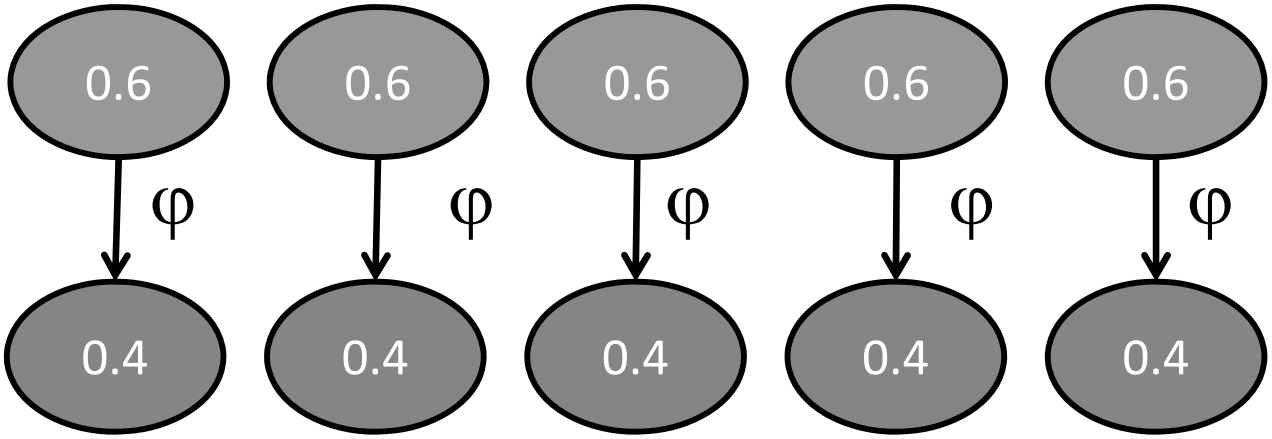
Esempio: S^* (2/7)

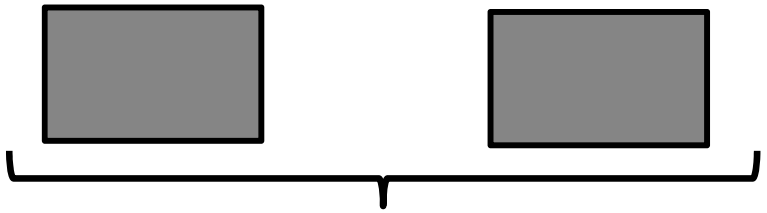
Ragioniamo sul duale …



$$m^* = n - m = 5 - 3 = 2 = k$$

Esempio: S^* (3/7)

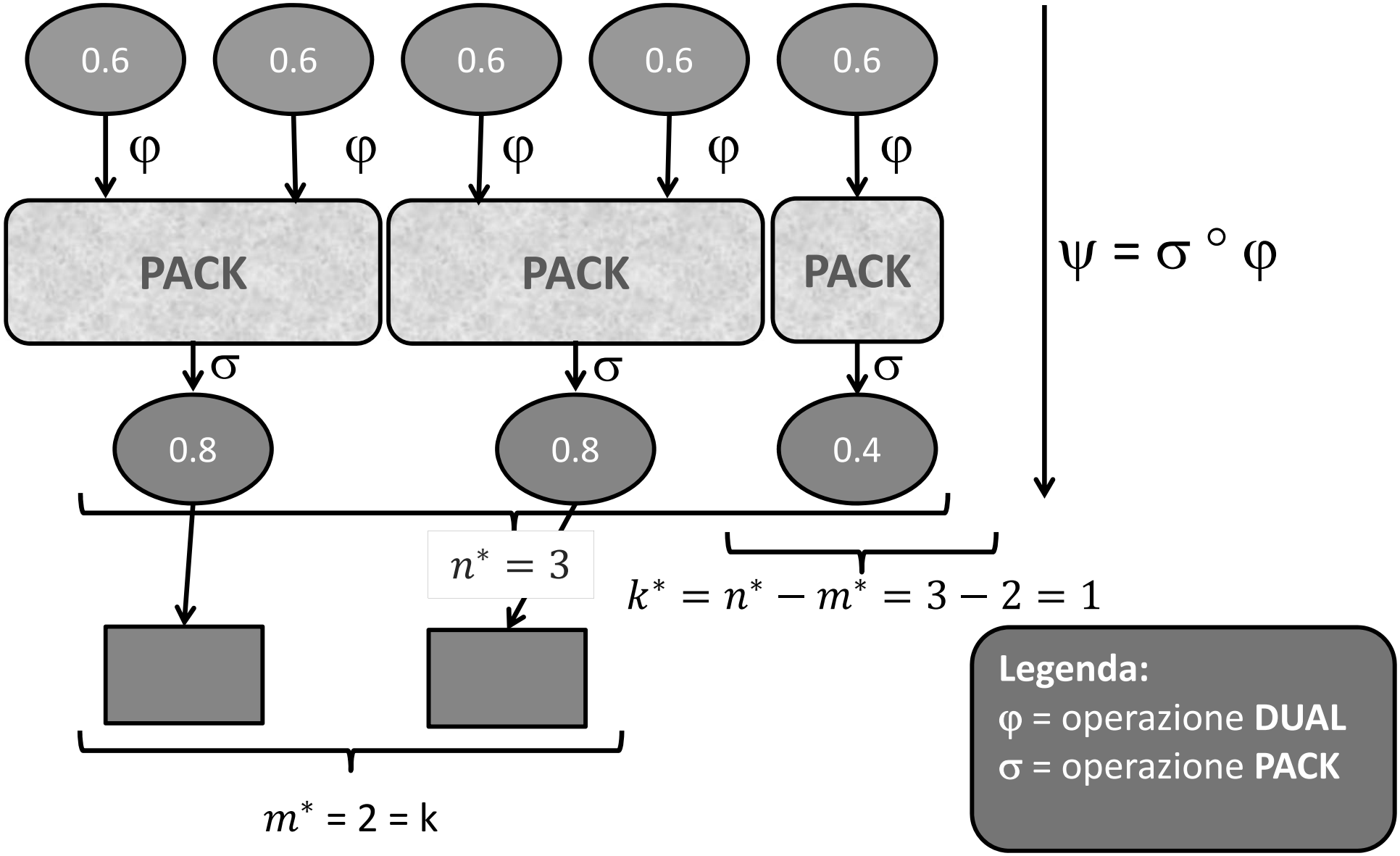




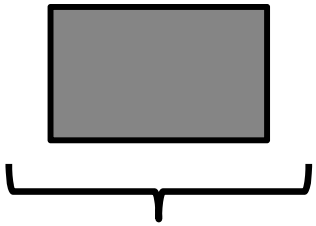
$$m^* = 2 = k$$

Legenda:
φ = operazione **DUAL**

Esempio: S^* (4/7)

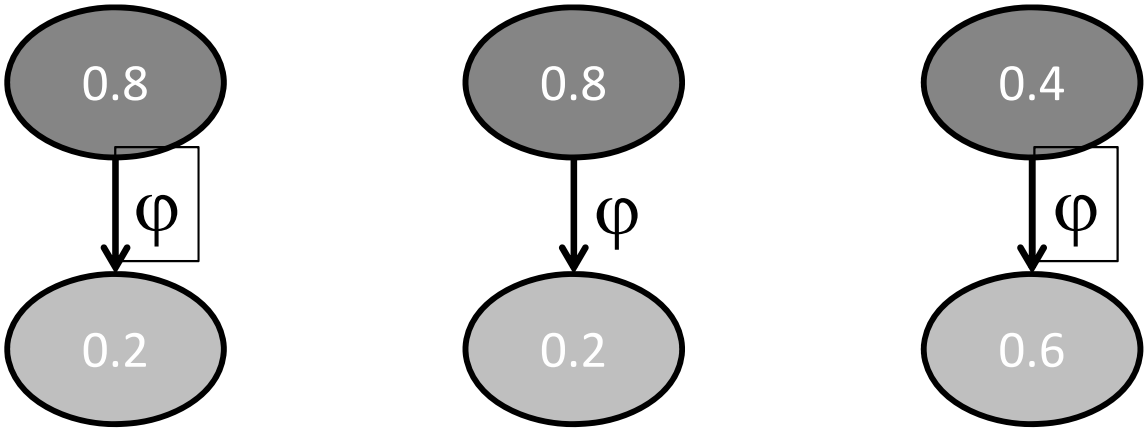

0.6
0.6
0.6
0.6
0.6
φ
φ
φ
φ
φ
φ
PACK
PACK
PACK
ψ = σ ° φ
σ
σ
σ
0.8
0.8
0.4
$n^* = 3$
$k^* = n^* - m^* = 3 - 2 = 1$
m^* = 2 = k
Legenda:
φ = operazione DUAL
σ = operazione PACK

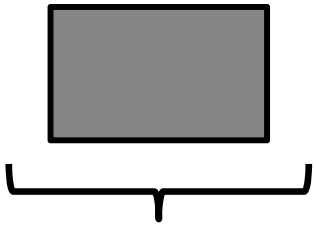
Esempio: S^{**} (5/7)



$m^{**} = 1 = k^{*}$

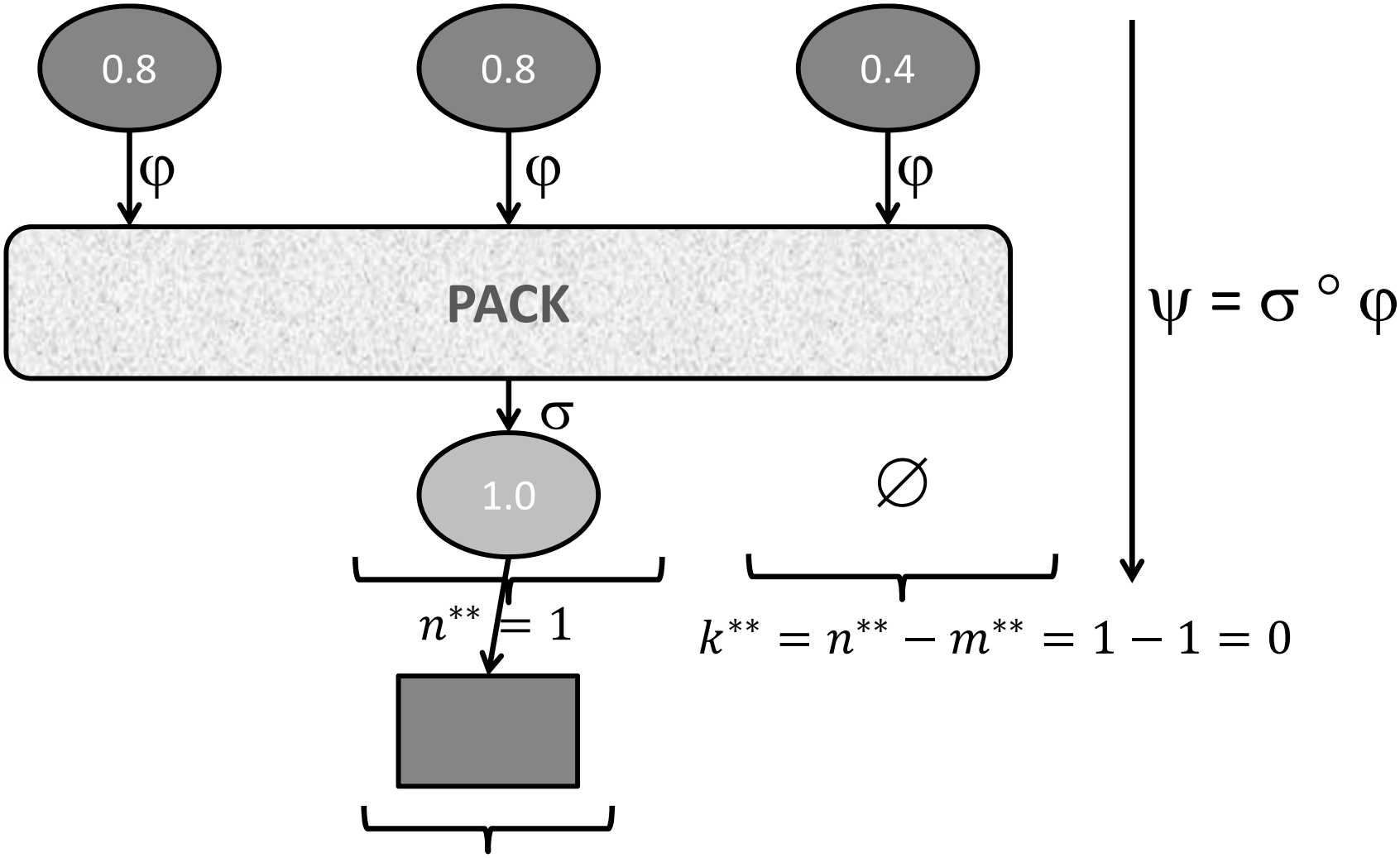
Esempio: S^{**} (6/7)





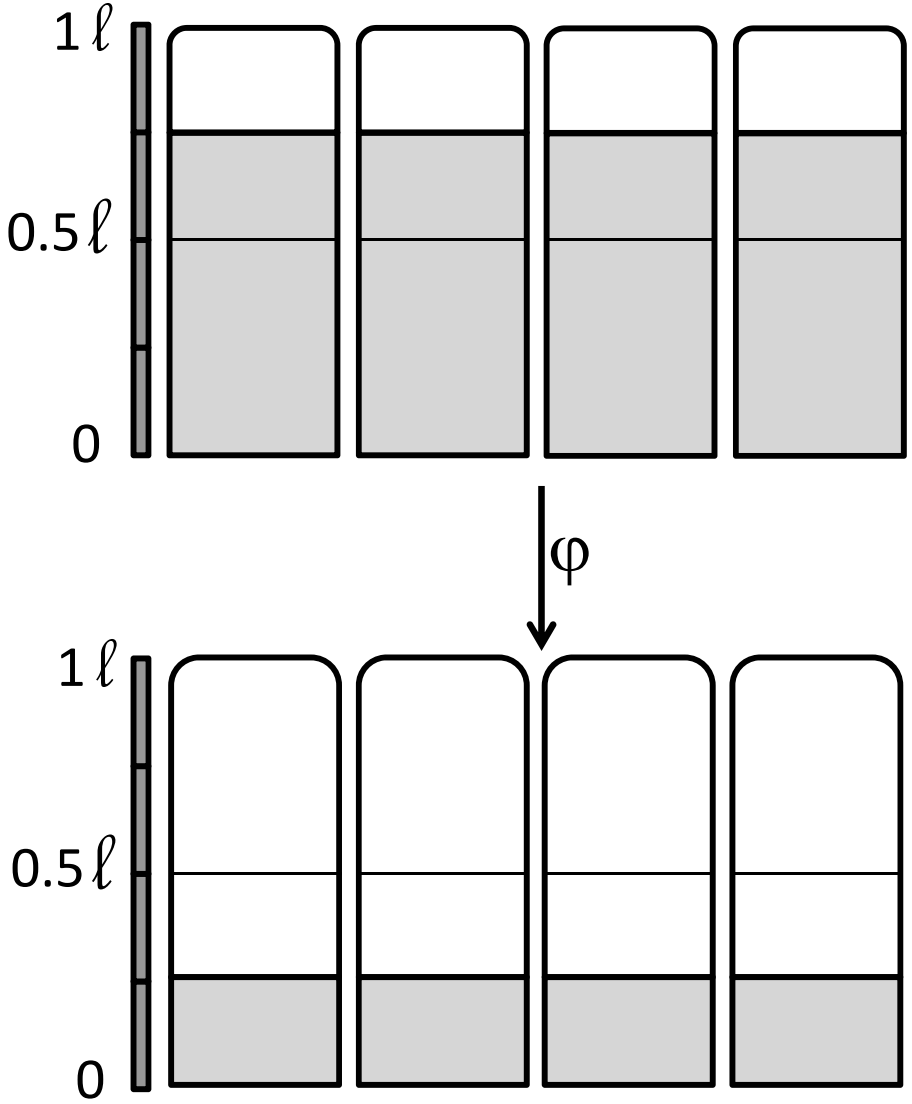
$$m^{**} = 1 = k^{*}$$

Esempio: S^{**} (7/7)



Algoritmo di riduzione: terminazione

Lemma: $\psi = \left| \sigma \circ \varphi \left(\bigcup_1^4 \tau_i \right) \right| \leq \left\lceil \frac{|\tau|+1}{2} \right\rceil$



Intuizione

$\sum_1^4 R = 3 \Rightarrow$ m = 3

n = 4

k = 1 < m

Nel sistema duale

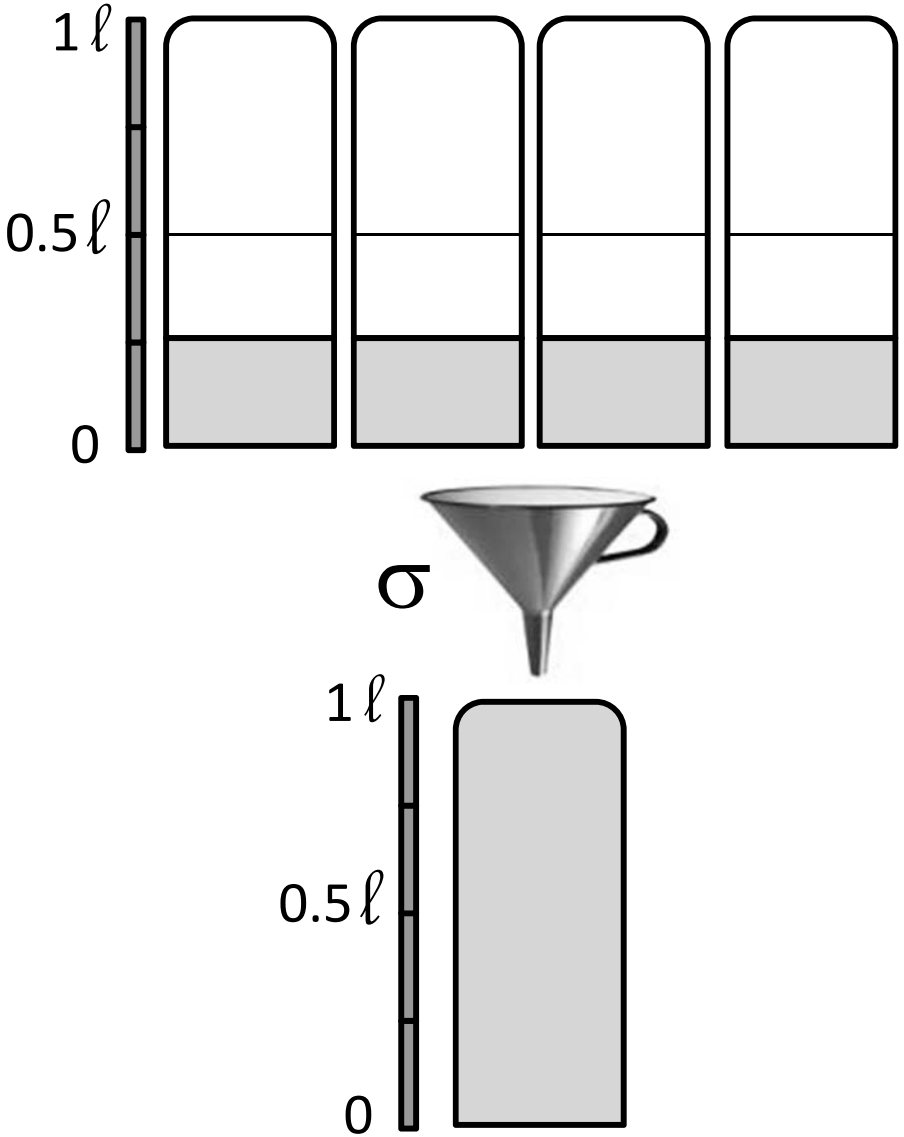
$\sum_1^4 R = 1 \Rightarrow m^* = 1 =$ **k**

$n^* = 1$

$k^* = 0$

Algoritmo di riduzione: terminazione

Lemma: $\psi = \left| \sigma \circ \varphi \left(\bigcup_1^4 \tau_i \right) \right| \leq \left\lceil \frac{|\tau|+1}{2} \right\rceil$



$\Rightarrow \psi$ -REDUCE ($\sigma \circ \varphi$) ha termine, perché a ogni passo si riduce il *workload* (*rate* complessivo), e quindi anche il numero di processori necessari, e con σ -PACK almeno dimezzo il numero di task

Teorema terminazione: dopo un numero finito ***p*** di iterazioni l'algoritmo termina riducendo a un sistema uniprocessore (con *rate* complessivo uguale a uno)

Passi fondamentali di RUN

Riassumendo:

- Operazione DUAL (φ)
- Operazione PACK (σ)
- Operazione REDUCE ($\psi = \sigma \circ \varphi$) riduce la dimensione del problema a ogni passo ($\sim$ dimezza)
- Teorema (validità del duale): Σ valido $\Leftrightarrow$ Σ^* valido
- Poiché ogni *task* duale rappresenta il tempo *idle* del *task* primario, trovare lo *schedule* del sistema duale è equivalente a trovare quello del sistema primario

Scheduling

L'algoritmo genera un albero dalle foglie alla radice (fase *off-line*)

Lo *schedule* si ricava ripercorrendo a ritroso l'albero generato dall'algoritmo e applicando ad ogni istante *t* le seguenti regole (fase *on-line*)

- ai passi di tipo φ si usa la regola del duale: esegui in $\Sigma(t)$ i *task* che non eseguono in $\Sigma^*(t)$
- ai passi di tipo σ (*pack*) si sceglie fra i *task* aggregati usando EDF

Il problema multiprocessore originario è stato quindi ricondotto a una collezione di problemi uniprocessore (**composizionalità** del problema)