Polynomial Time Algorithms for Minimum Energy Scheduling

Ph.Baptiste M.Chrobak C.Dürr
CNRS UCR CNRS

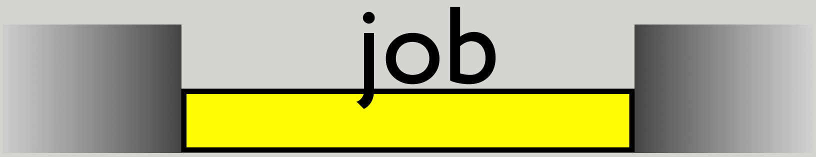
lip6 JFRO 2010

Why Minimize Energy?

- In 2005 US server farms consumed 1.2% of all electricity
- Please, turn off computers when not needed
- Our objective : schedule jobs such that we need to turn the machine on fewest possible

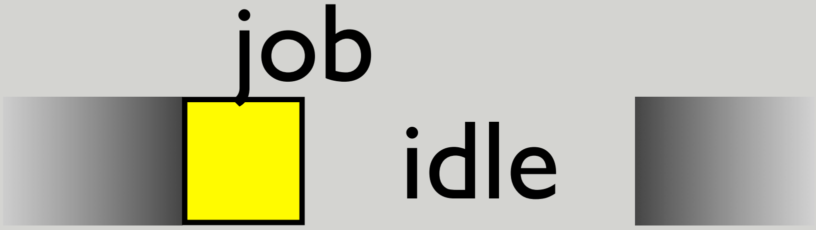
Two models

speed scaling



if speed is s
power consumption is $s^{\alpha}+c$
for some $\alpha \in [2,3]$
and some leaking c.

sleep states

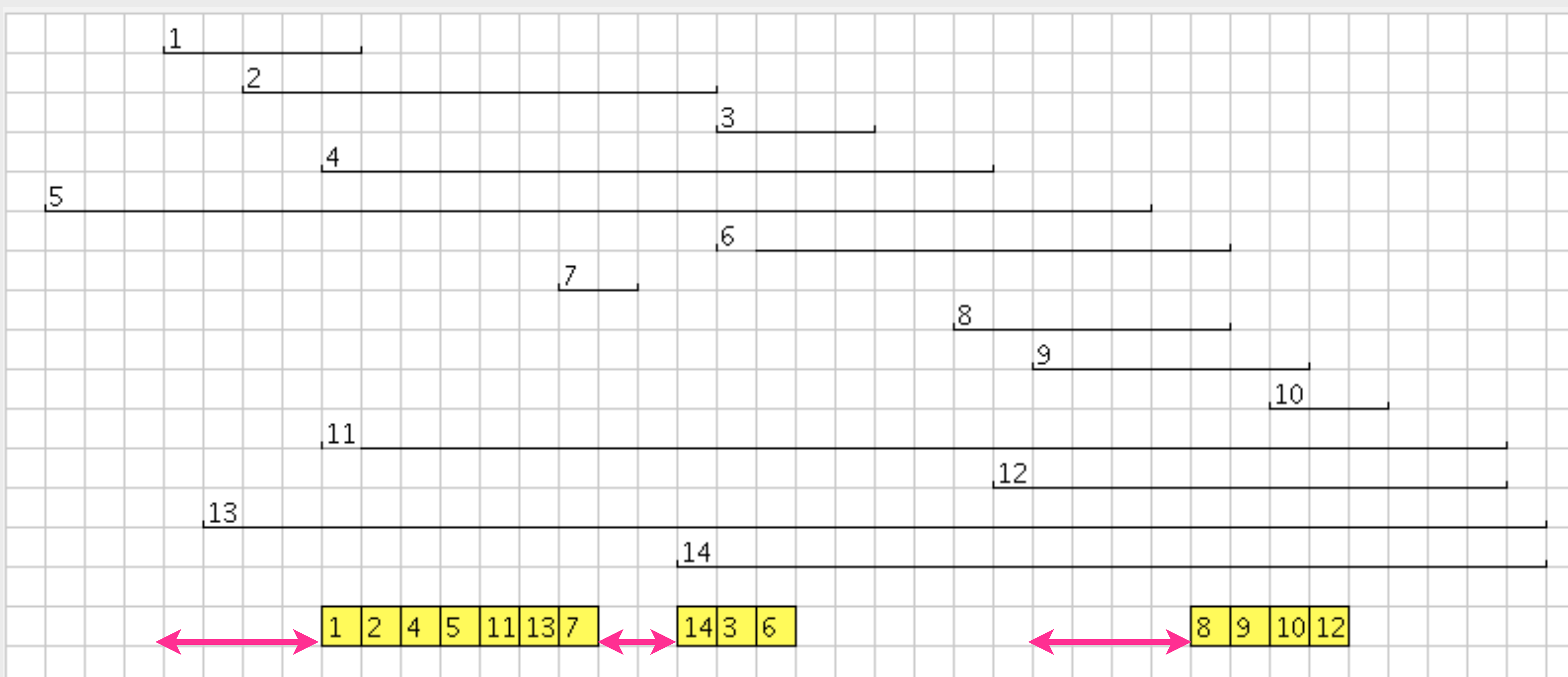


during idle time power consumption is 0, except for the L last units, necessary for wakeup.

Our Problem

- Input : *switch-on-time* L, n jobs with length p_i and release time, deadline $[r_i,d_i]$
- Output feasible preemptive schedule with min cost
- Cost of each idle interval [s,t] is min{L,t-s}

example
p_i=1
L=4



Our algorithms

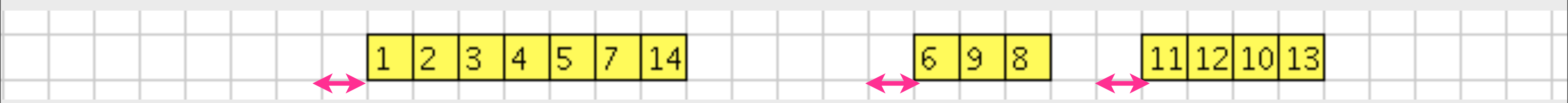
	p_i=1	general p_i
L=1	$O(n^4)$	$O(n^5)$
general L	$O(n^4)$	$O(n^5)$

- when different speeds are allowed
  open but there is a 2-approx. [Irani,Shukla,Gupta,SODA'03]
- when jobs arrive on-line
  needs a different model [Augustine,Irani,Swamy,FOCS'04]

Our algorithms

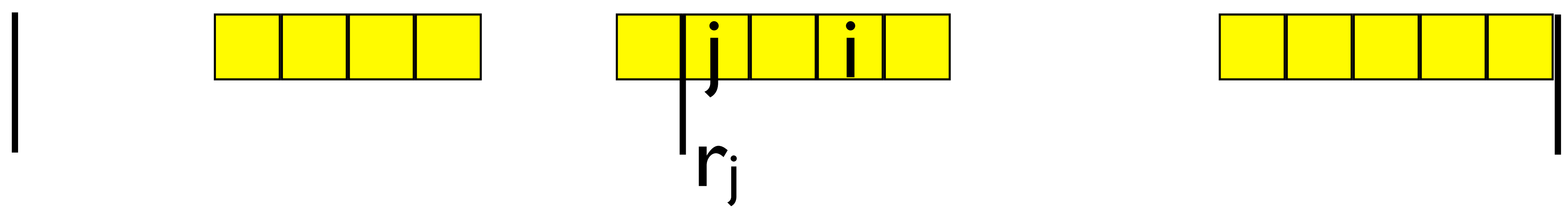
	$p_i=1$	general p_i
$L=1$	$O(n^4)$ previously $O(n^7)$	$O(n^5)$
general L	$O(n^4)$	$O(n^5)$

[Baptiste,SODA'06]



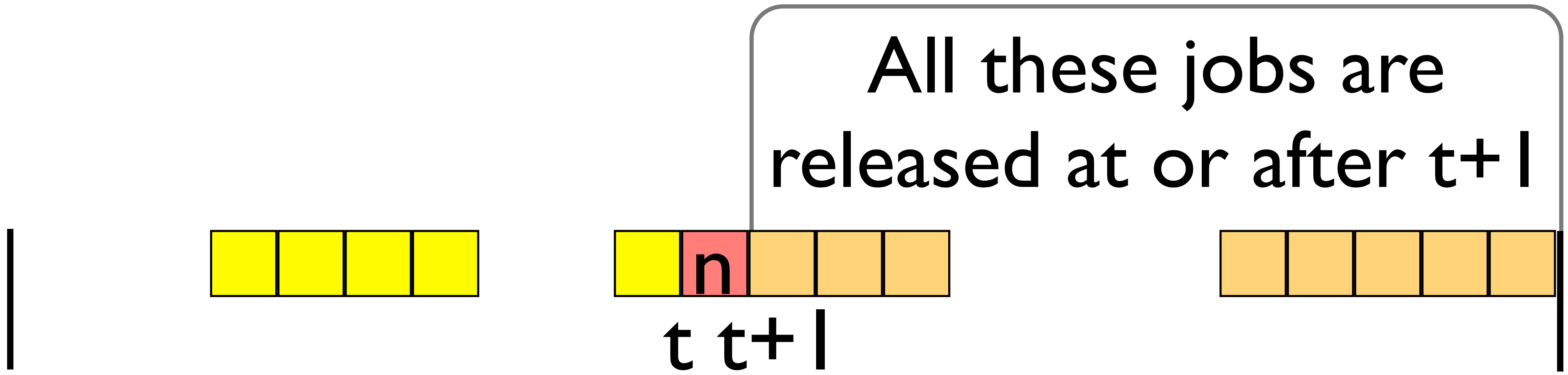
= Packing unit size jobs in blocks with least number of gaps

Obs1: execution times



- Without loss of generality every block contains a job starting at its release time
- So all starting times are of the form T:=$\{r_j+a : j=1,\dots,n, -n<a<n\}$ and there are $O(n^2)$ of them

Obs2: execution order



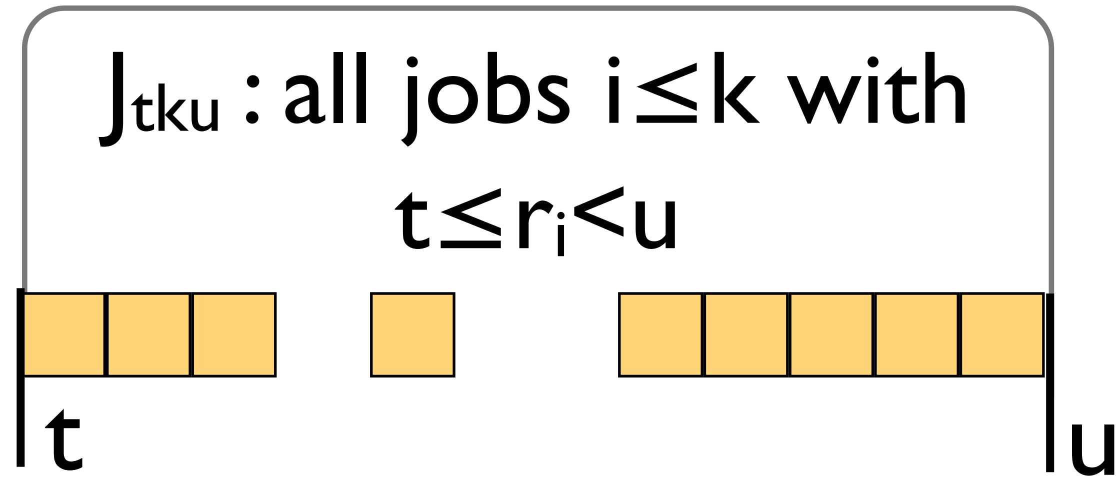
By an exchange argument, we can assume that the jobs are scheduled in the blocks with the Earliest Deadline Policy. (assume $d_1 \leq d_2 \leq \ldots \leq d_n$)

Basically we guess where the least urgent job n is scheduled, it divides the schedule into independent sub-schedules.

Sub-problem

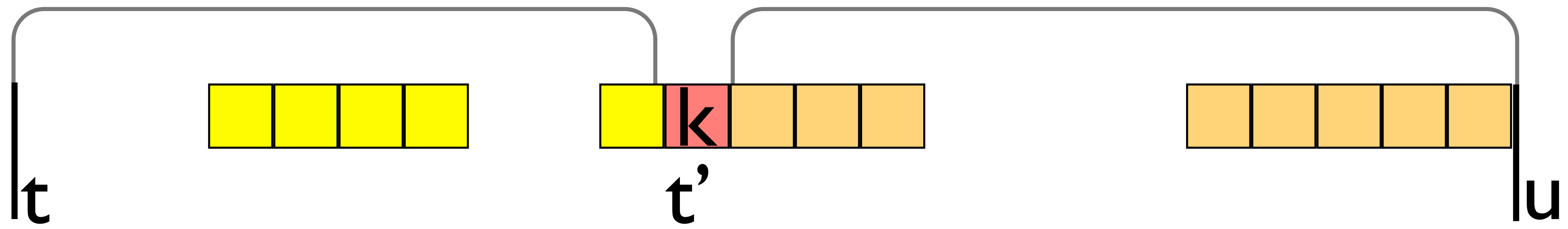
$t, u \in T$

$k \in \{1, .., n\}$



Def: G_{tku} = minimum number of gaps over all schedules of J_{tku}
(including gaps between t and first block and between last block and u)

Recursion



Basis: if $J_{tku}=\{\}$, $G_{tku}=0$ when $t=u$ o/w $G_{tku}=1$

Basis: if $k \notin J_{tku}$, $G_{tku}=G_{t,k-1,u}$

Induction: $G_{tku}=\min\{G_{t,k-1,t'} + G_{t'+1,k-1,u} : t \leq t' < u\}$

Complexity $O(n^7)$:
$O(n^5)$ variables, minimize over $O(n^2)$ choices

The inversion trick

G_{tku} has O(n) values, while u ranges over O(n^2) values

range of u

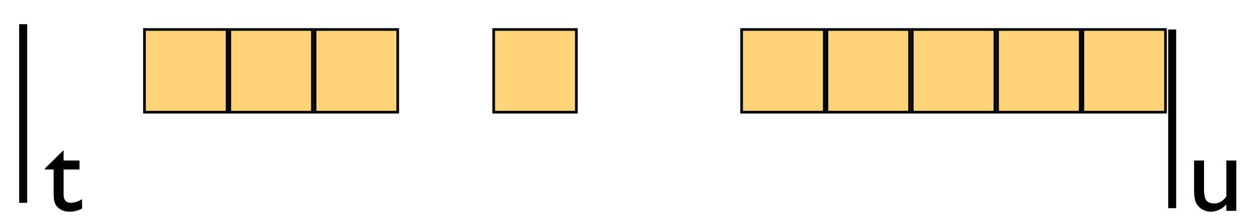
range of g

0100100
0110110

1 means that there is a schedule of J_{tku} with at most g gaps

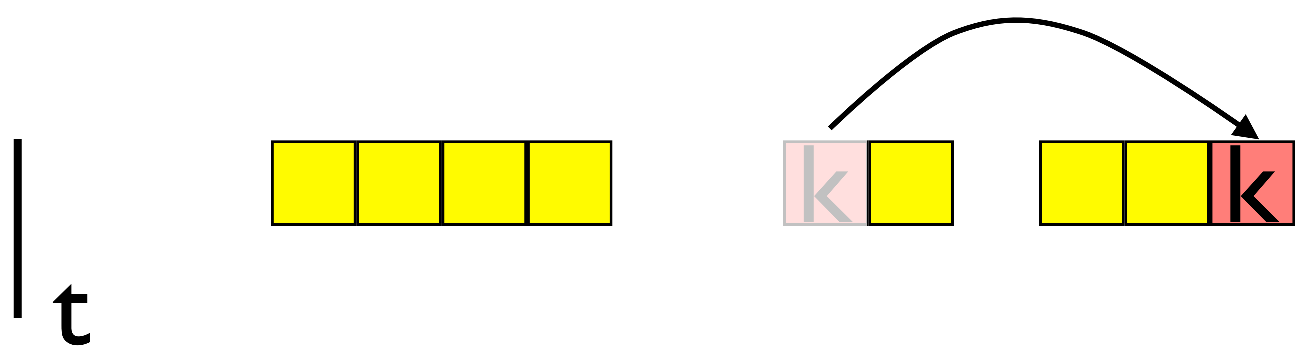
Invert: U_{tkg} = maximal u such that there is a schedule of J_{tku} of at most g gaps and makespan u

Only O(n4) variables



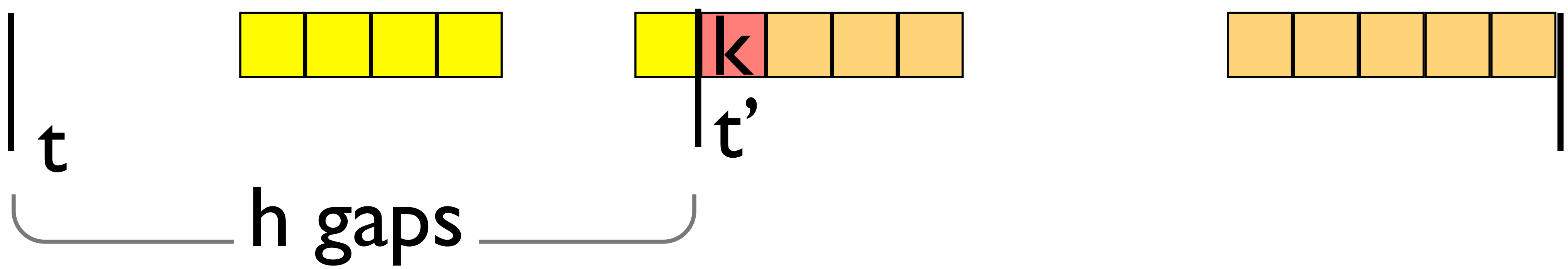
Obs3: completion times

- If $d_i=d_j$, $r_i \le r_j$, we can assume that i completes before j and set $d_i:=d_i-1$
- Without loss of generality we can assume $d1<d2<...<dn$

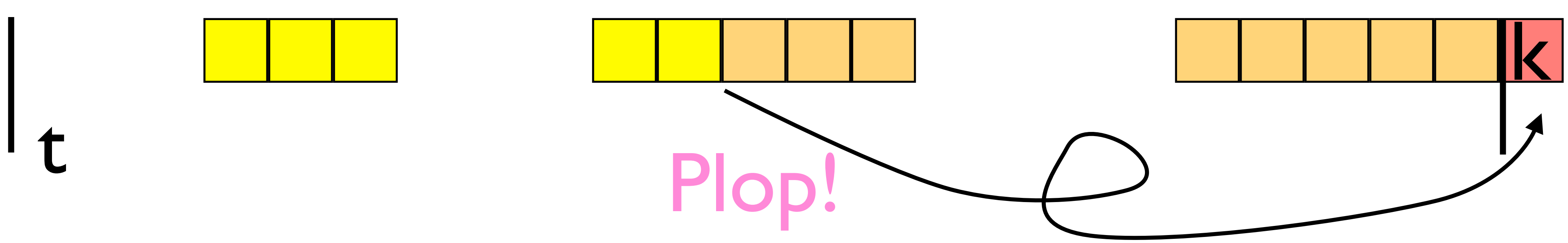


- Therefore if k is in the optimal schedule for U_{tkg}, then either inside a block or at the end

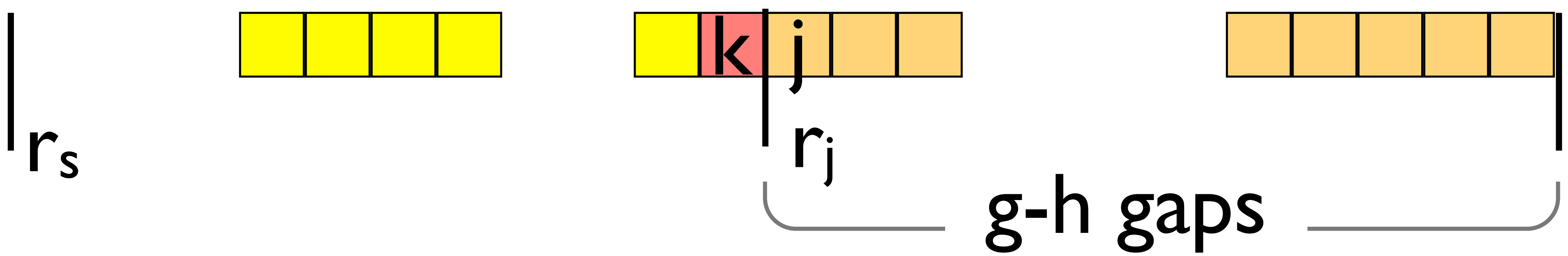
Case job k is scheduled inside a block



- k starts at $U_{t,k-1,h}$ for some $h \leq g$ otherwise if $t' < U_{t,k-1,h}$ we could replace...



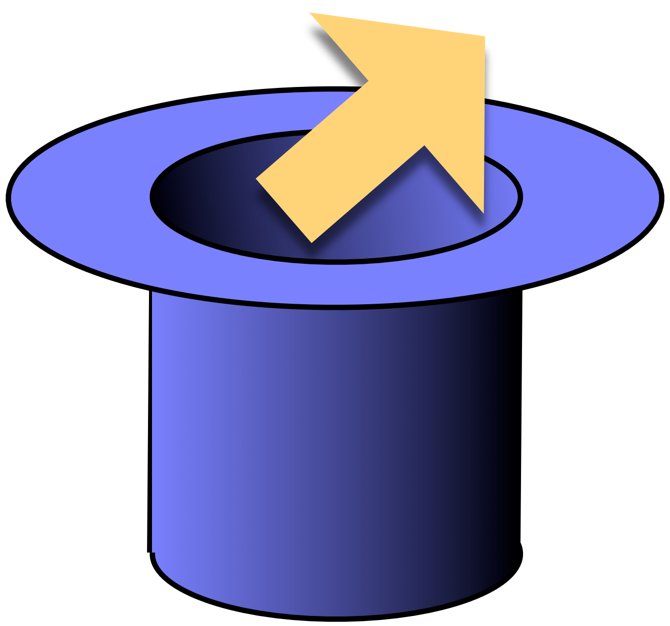
Obs4: t is always some release time



- So compute rather U_{skg} = max u such that there is a $J_{rs,k,u}$ schedule with at most g gaps and makespan u
- Only $O(n^3)$ variables

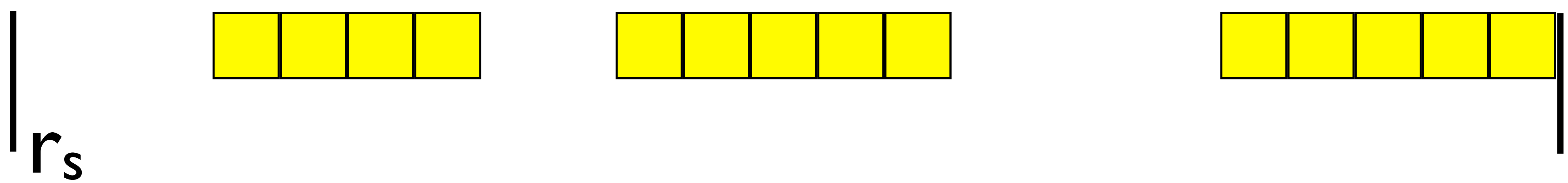
The dynamical program

U_{skg} = max {... some choice ....}



- For a proof of correctness fix some schedule realizing U_{skg}, and go through cases ...
- Where is k scheduled?

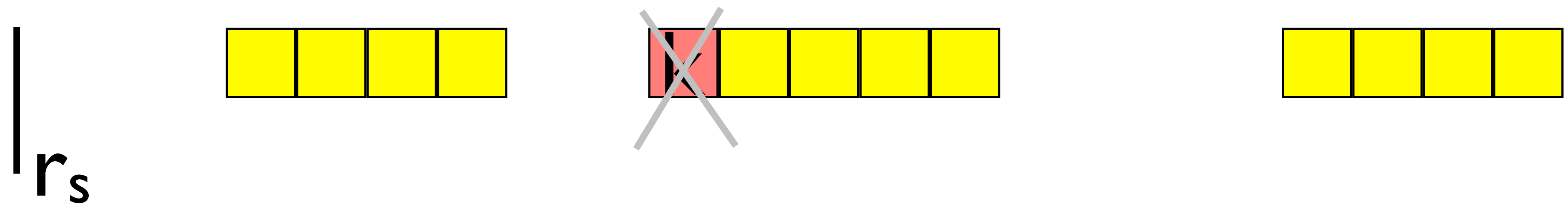
Job k might not be scheduled at all



Then $U_{skg} = U_{s,k-1,g}$

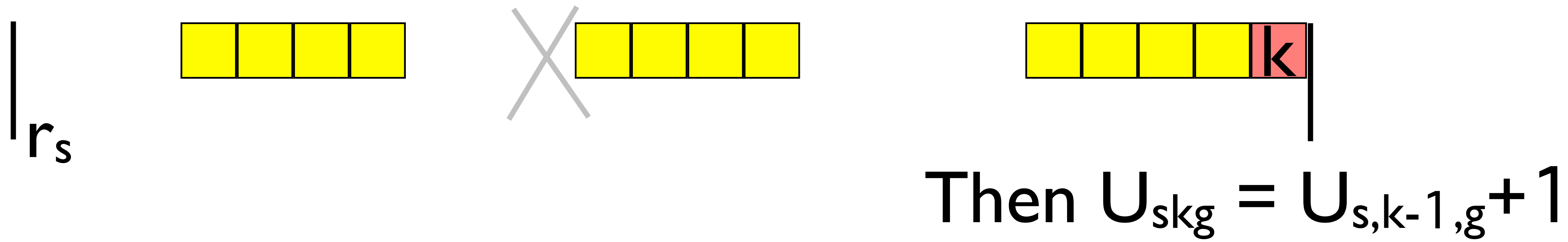
- Boring!

Job k might be scheduled at the border of a block



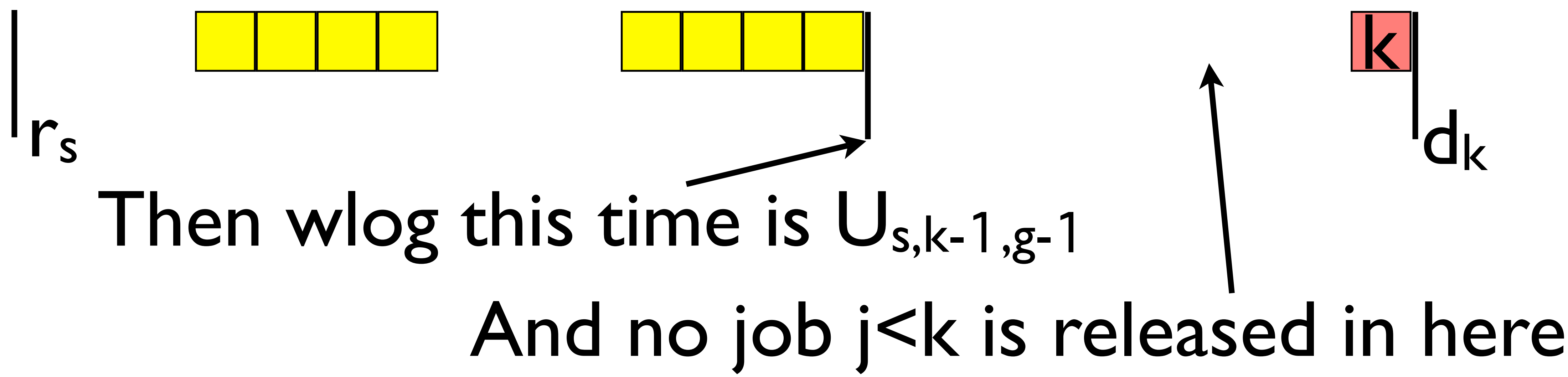
- Then it must be at the end of the last block
- Moving k to the end is valid since $d_1 < \ldots < d_n$

Job k might be scheduled at the border of a block

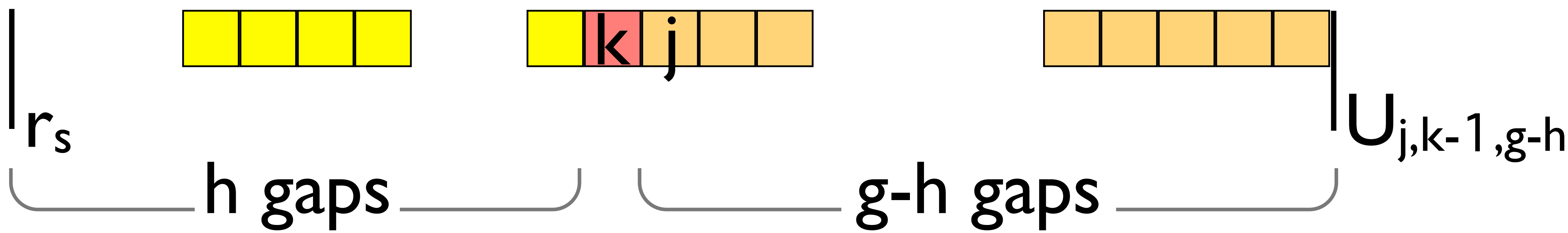


- Then it must be at the end of the last block
- Moving k to the end is valid since $d_1<\ldots<d_n$

Job k might be scheduled alone in a block



Job k might be scheduled inside a block

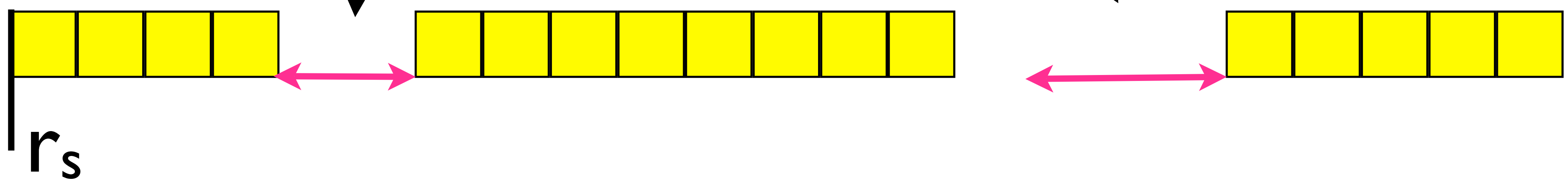


- Then k is scheduled at a time $U_{s,k-1,h}$ for some h and there is a job j with $r_j = U_{s,k-1,h}+1$
- In overall U_{skg} is the maximization over O(n) choices
  - Algorithm in time $O(n^4)$

Our algorithms

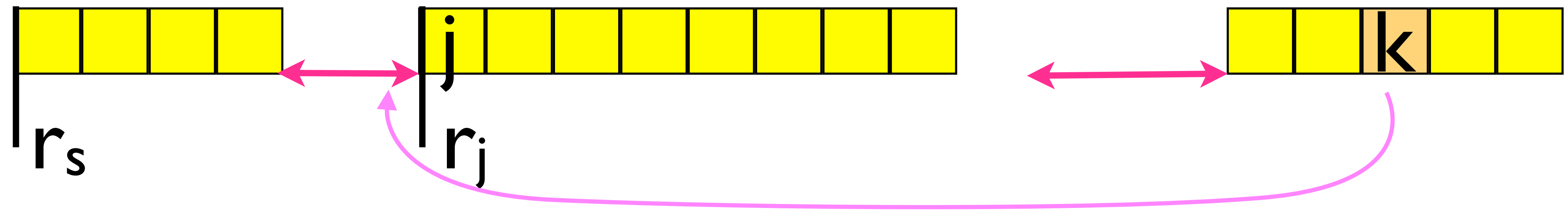
	$p_i=1$	general p_i
L=1	$O(n^4)$ previously $O(n^7)$	$O(n^5)$
general L	$O(n^4)$	$O(n^5)$

Small / large gaps



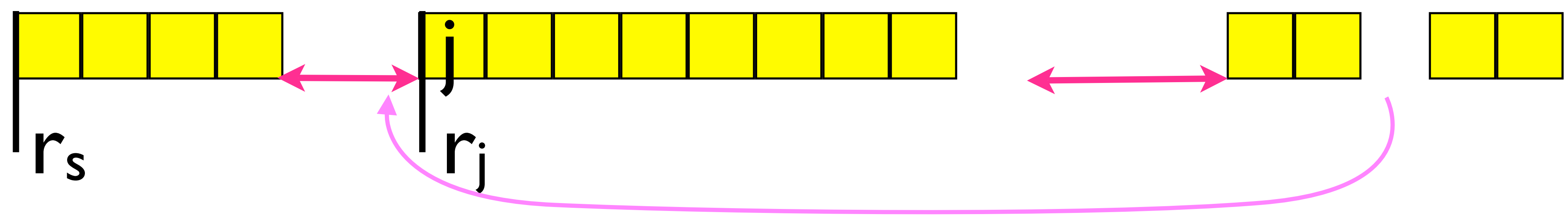
- Small gaps cost their length
- Large gaps cost L

A decomposition



- Wlog jobs scheduled after a small gap, are released after it
- Otherwise we can create a more *dominant* schedule

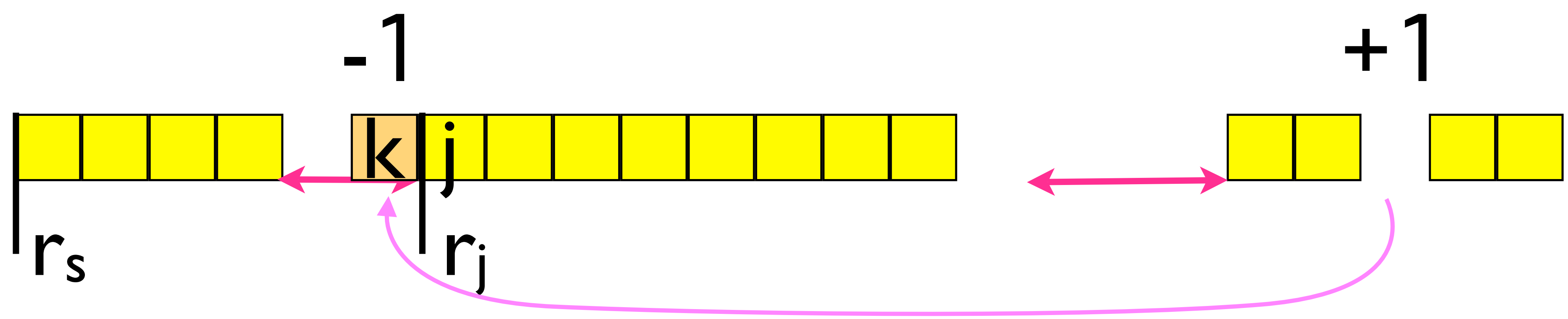
A decomposition



- Wlog jobs scheduled after a small gap, are released after it
- Otherwise we can create a more *dominant* schedule

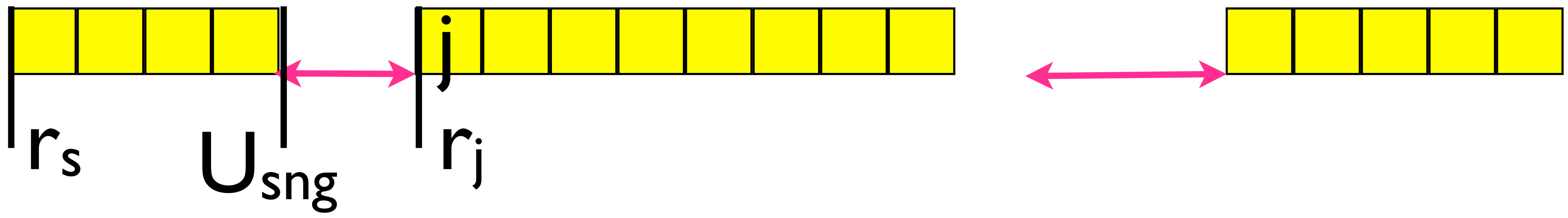
A decomposition

Change in cost :



- Wlog jobs scheduled after a small gap, are released after it
- Otherwise we can create a more *dominant* schedule

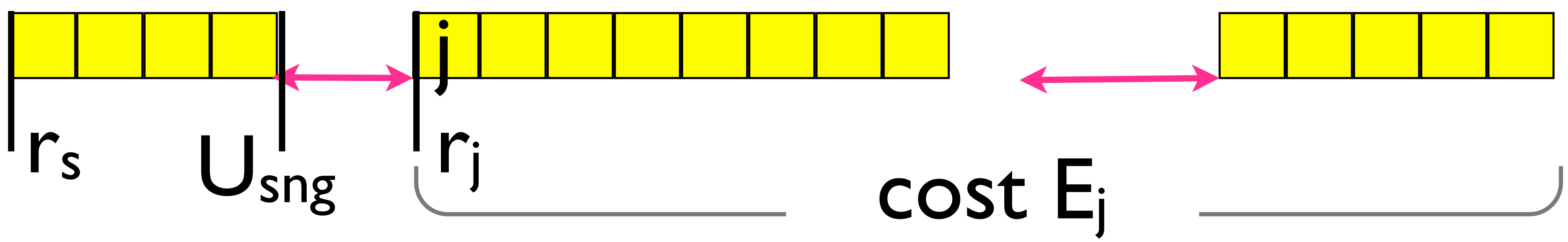
A dynamic program



- Def: E_s=min cost of a schedule with all jobs j, $r_j \geq r_s$
- Guess g, the number of large gaps before the first small gap.

A dynamic program

$$E_s \leftarrow \min_{0\le g\le n} \begin{cases} Lg & \text{if } U_{n,s,g} > \max_j r_j \\ Lg + r_l - u + E_l & \text{otherwise, where } u = U_{n,s,g}, r_l = \min\{r_j : r_j > u\} \end{cases}$$



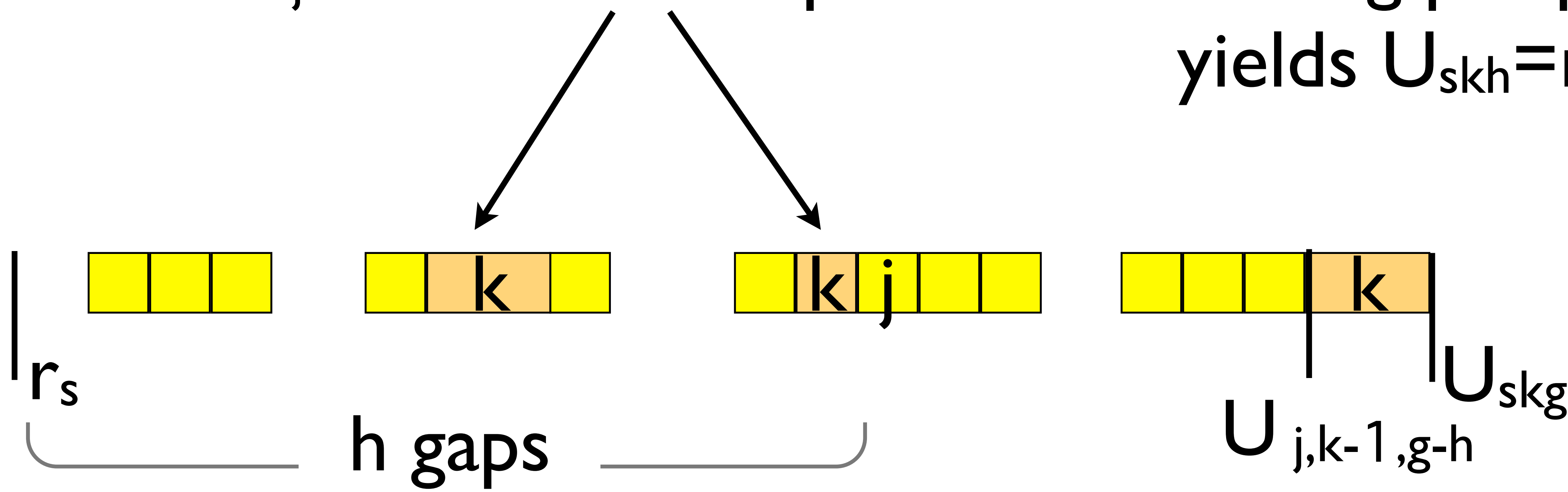
- Def: E_s=min cost of a schedule with all jobs j, $r_j \geq r_s$
- Guess g, the number of large gaps before the first small gap.

Our algorithms

	$p_i=1$	general p_i
$L=1$	$O(n^4)$ previously $O(n^7)$	$O(n^5)$
general L	$O(n^4)$	$O(n^5)$

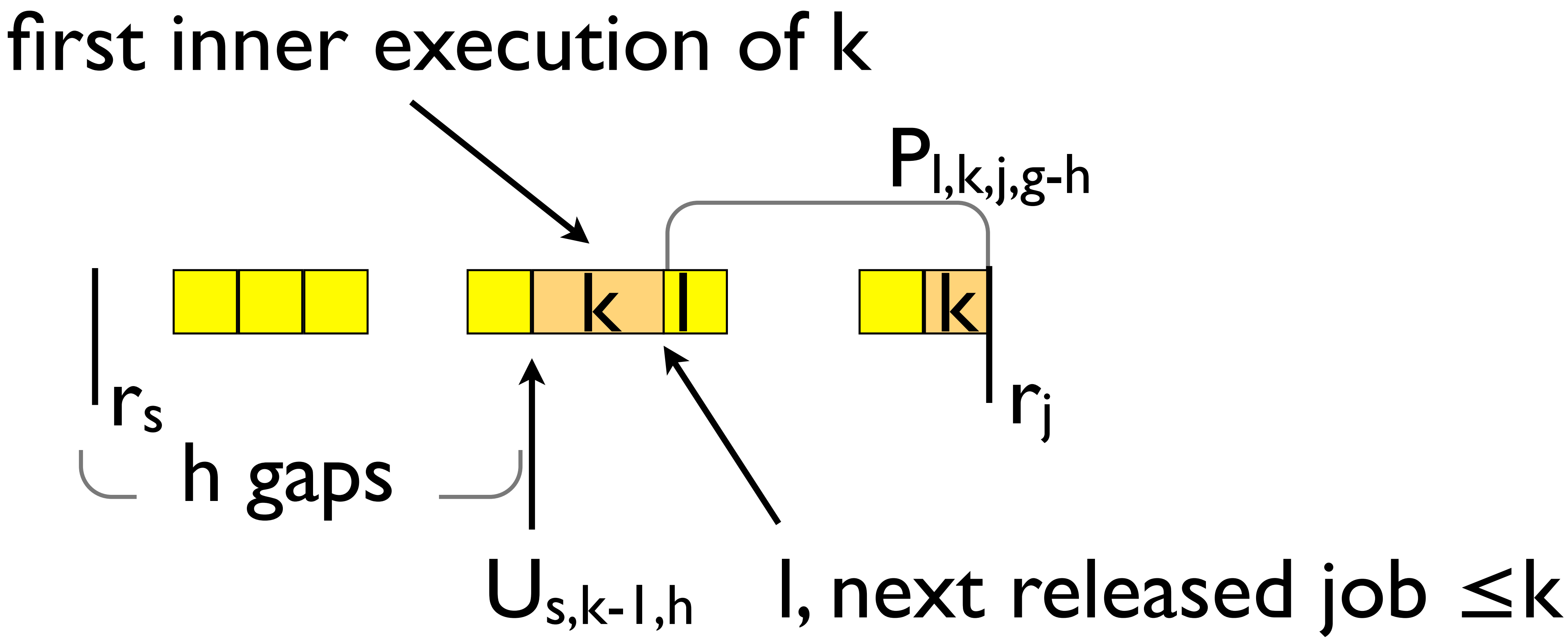
General Idea

Def: P_{skjh}=min. amount p such that setting p_k:=p, yields $U_{skh}=r_j$



Then $U_{skg} = U_{j,k-1,g-h} + p_k - P_{skjh}$

Decompostion for P_{skjg}



Open Problems

- Allow identical parallel machines
  (SOLVED: Demaine et al. SPAA'07)
- Allow several power down states
  [Adriana Lopez generalized our algorithms]
- Allow several processor speeds
  [in progress]