Energy-Efficient Scheduling

Susanne Albers
University of Freiburg
Germany

Motivation

- Energy consumption grows exponentially in computing devices
  computers, embedded systems, portable devices, ...
- Performance of processors doubles every three years

- Critical in battery-operated devices
- Critical in terms of cost (computer centers)
- Critical as most energy is converted into heat

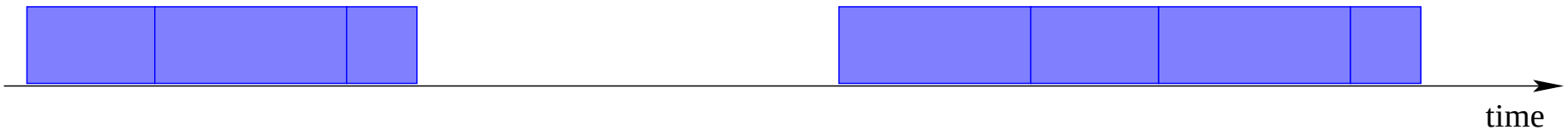
Algorithmic techniques

- Power-down strategies: Put system into sleep state when idle.

- Dynamic speed scaling: Microprocessors can run at variable speed, e.g. Intel XScale, Intel Speed Step, AMD PowerNow

Power-down strategies

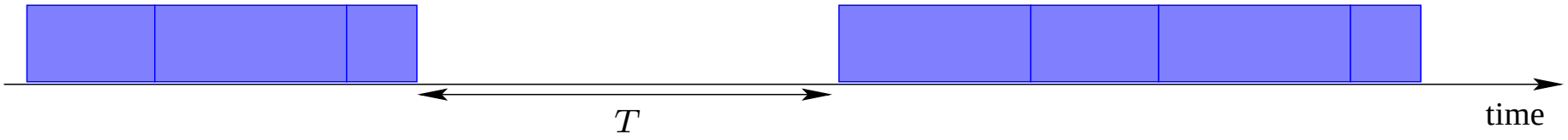
- Active mode: 1 energy unit per time unit.
- Sleep mode: 0 energy units per time unit.
- Wake-up operation: W energy units.
- When new jobs arrives, system must be (moved) to active mode.



Deterministic online algorithm

ALG: Power down after W time units.

Thm: ALG is 2-competitive.



- $T \leq W$

  $E(ALG) = T$ $\qquad$ $E(OPT) = T$

- $T > W$

  $E(ALG) = W + W = 2W$ $\qquad$ $E(OPT) = W$

Randomized online algorithm

RALG: Power down after t time units with probability

$$p_t = Ke^{t/W}$$

$$K = 1/(W(e-1))$$

Thm: RALG achieves competitive ratio of $e/(e-1) \approx 1.58$.

General setting

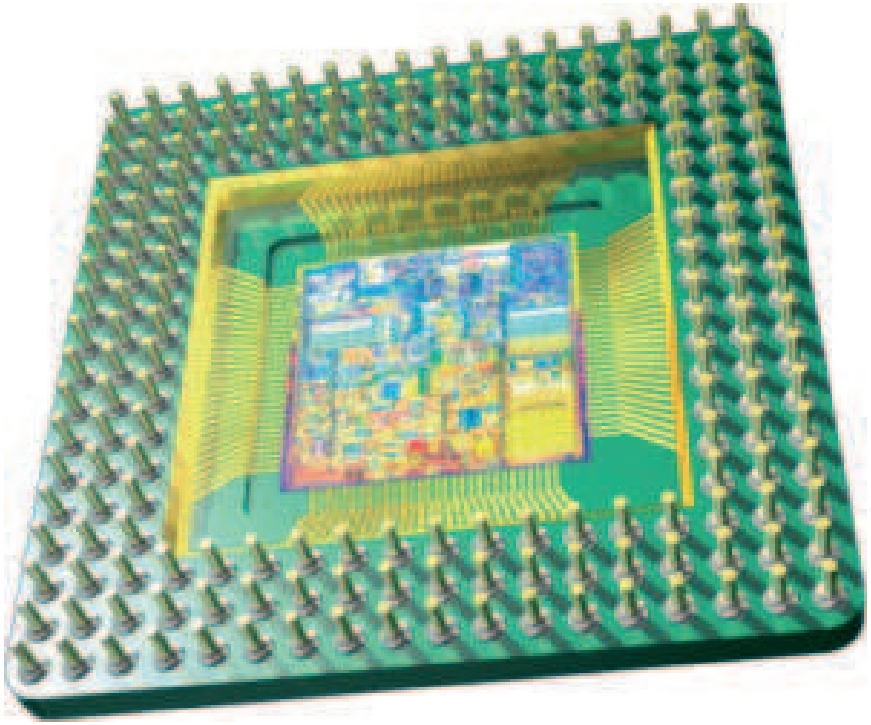
Augustine, Irani, Swamy FOCS 2004

- States $s_0, s_1, \ldots, s_m$ with power consumptions $k_0 > k_1 > \ldots > k_m$ per time unit
- d_{ij} : transition cost from s_i to s_j

Results:

- 5.8-competitive simple deterministic online algorithm.
- $(c + \epsilon)$-competitive algorithm, c best ratio for a given system.
- Good algorithms when idle periods are governed by probability distributions.

Dynamic speed scaling



The higher the speed, the higher the power consumption

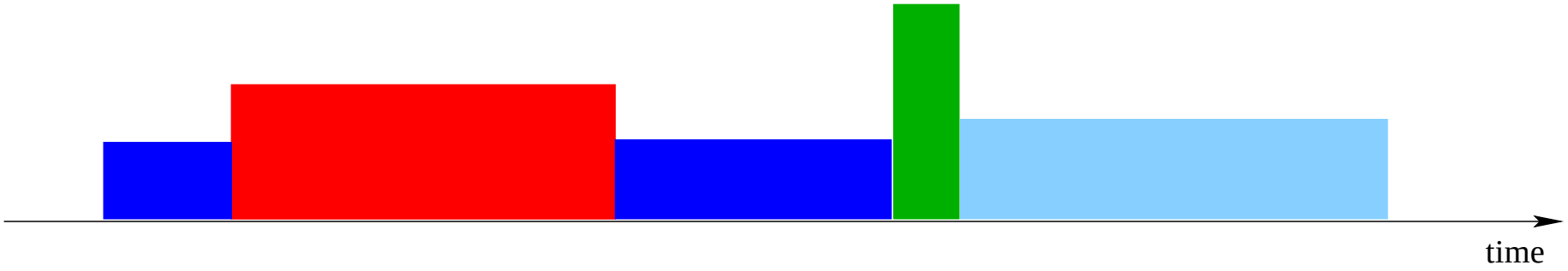
Speed s

Power consumption

$P(s) = s^{\alpha} \quad \alpha > 1$

Previous work

Deadline-based scheduling 1 processor



- Speed s Energy consumption $P(s) = s^{\alpha}$ $\alpha > 1$
- $\sigma = J_1, \ldots, J_n$

  J_i: a_i = arrival time

  b_i = deadline

  p_i = processing volume $t = p_i/s$
- Preemption allowed
- Construct feasible schedule minimizing total energy consumption.

Previous work

Offline setting

- polynomially solvable
  (Yao, Demers, Shenker 1995)

Online setting

- Average Rate: $\alpha^\alpha \leq c \leq 2^\alpha \alpha^\alpha$ (Yao, Demers, Shenker 1995)
  Optimal Available: $c = \alpha^\alpha$ (Bansal, Kimbrel, Pruhs 2004)
- Upper bound $2(\frac{\alpha}{\alpha-1})^\alpha e^\alpha$
  Lower bound $\Omega((\frac{4}{3})^\alpha)$
  (Bansal, Kimbrel, Pruhs 2004)

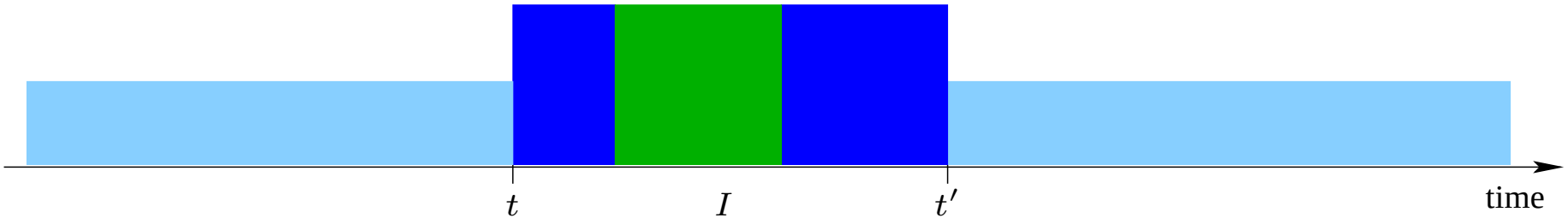
Optimal offline algorithm

- Density of interval I

$$\Delta_I = \frac{1}{|I|} \sum_{[a_i, b_i] \subseteq I} p_i$$

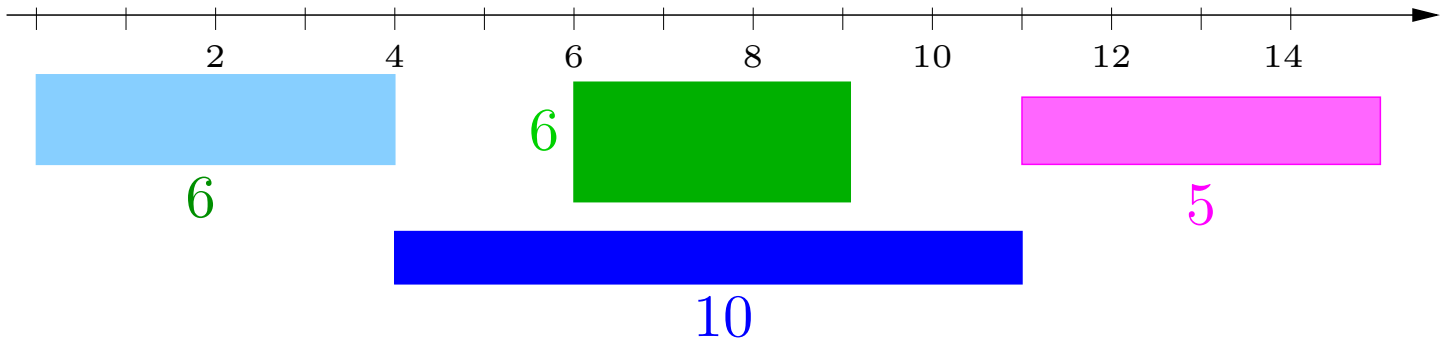
- Repeatedly determine interval I of maximum density.
  Schedule jobs in I at speed Δ_I using Earliest Deadline First
  Reduce problem instance by I.

Optimal offline algorithm

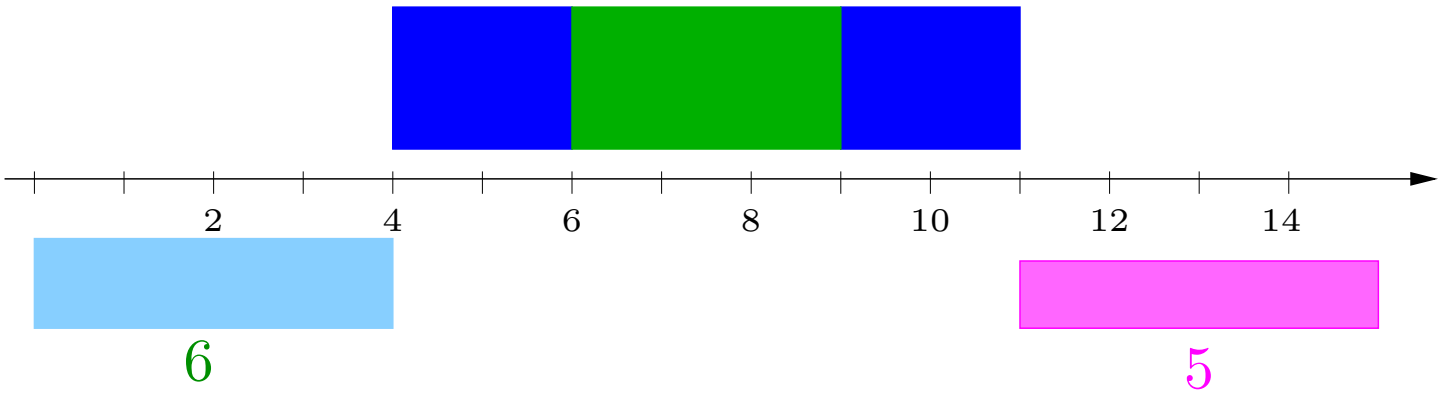


- $a_i \in I \quad \Longrightarrow \quad a'_i = t'$
- $b_i \in I \quad \Longrightarrow \quad b'_i = t$
- reduce arrival times / deadlines of at least t by $t' - t$

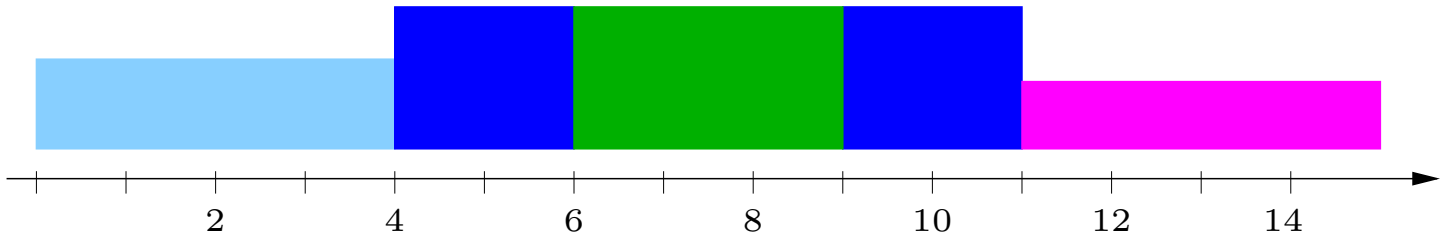
Example



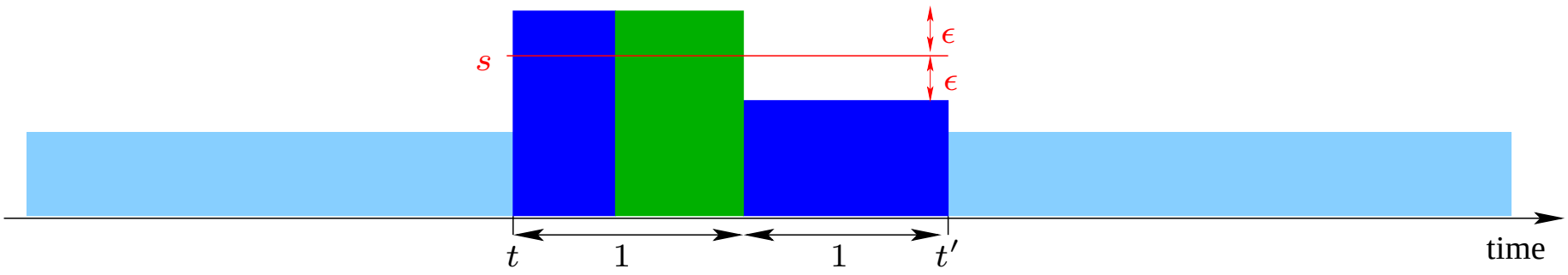
Example



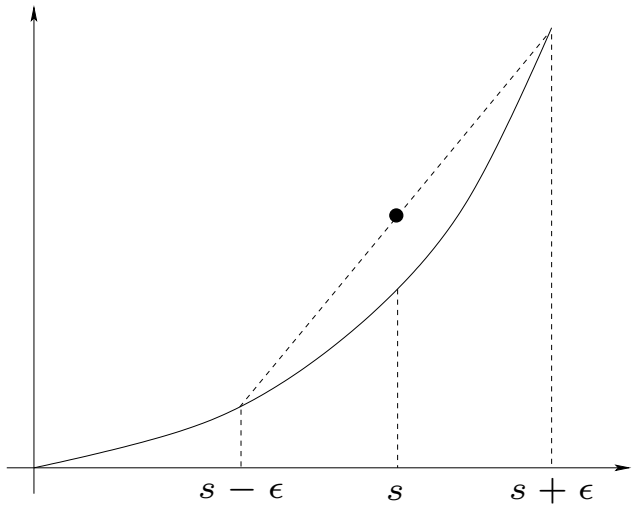
Example



Optimality proof



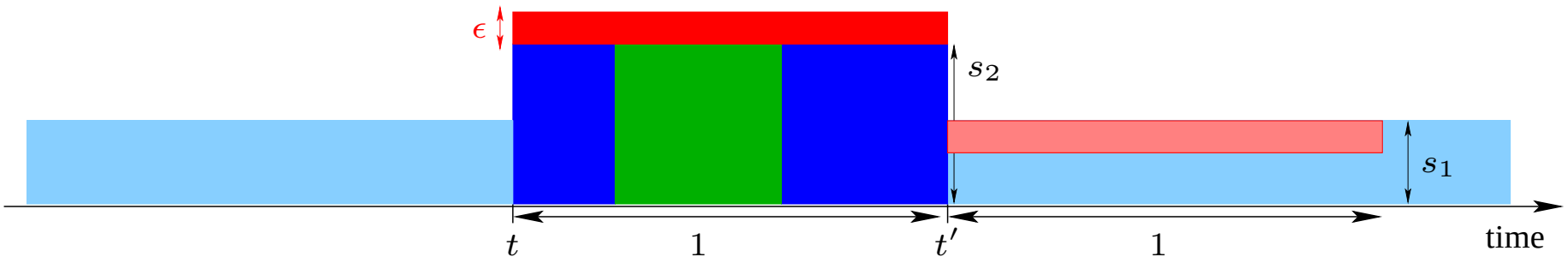
Use same speed throughout I.



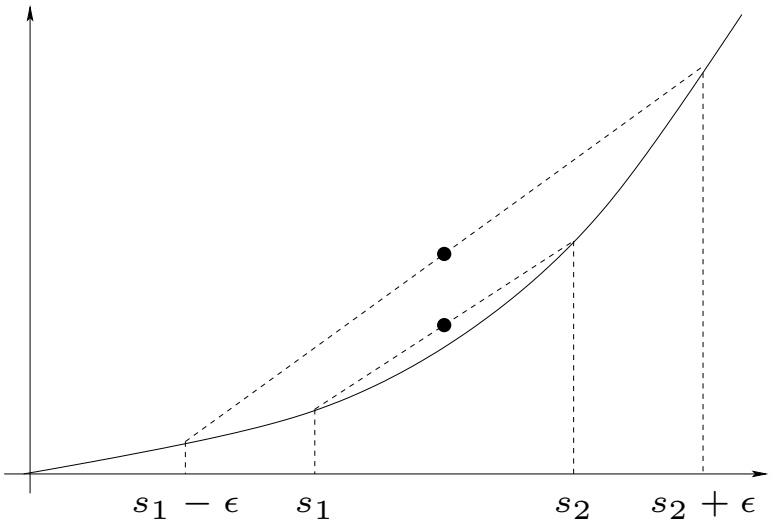
$$(s-\epsilon)^\alpha + (s+\epsilon)^\alpha > 2s^\alpha$$

$$\tfrac{1}{2}(s-\epsilon)^\alpha + \tfrac{1}{2}(s+\epsilon)^\alpha > s^\alpha$$

Optimality proof



Do not use speed higher than Δ_I in I.

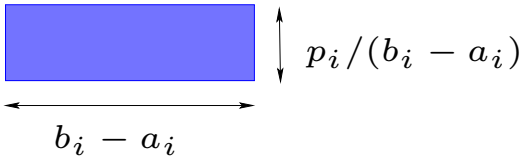


$$(s_1 - \epsilon)^\alpha + (s_2 + \epsilon)^\alpha > s_1^\alpha + s_2^\alpha$$

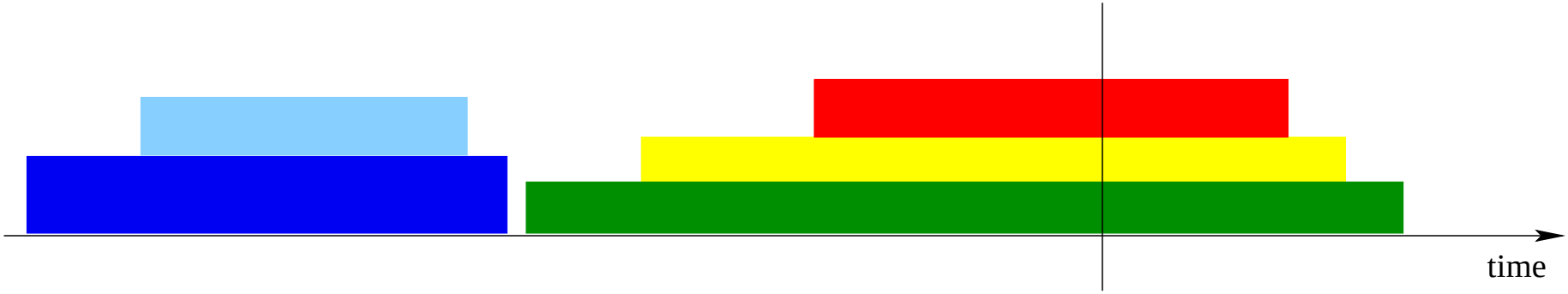
$$\frac{1}{2}\left((s_1 - \epsilon)^\alpha + (s_2 + \epsilon)^\alpha\right) > \frac{1}{2}\left(s_1^\alpha + s_2^\alpha\right)$$

Average Rate

1. Job density $\delta_i = p_i/(b_i - a_i)$

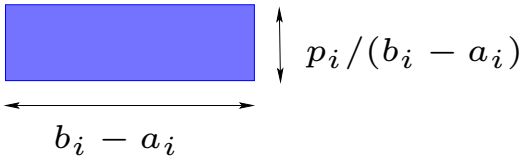


2. $s(t) = \sum_{i\,:\,t \in [a_i, b_i]} \delta_i$

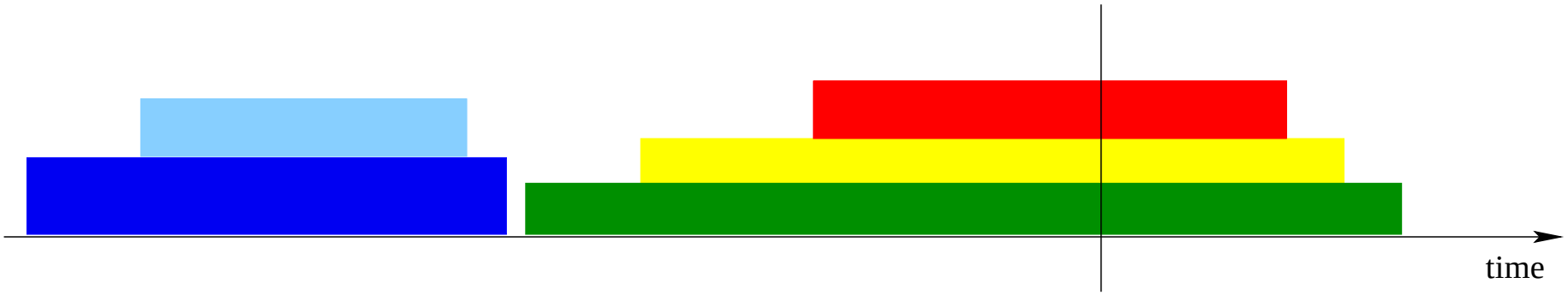


Average Rate

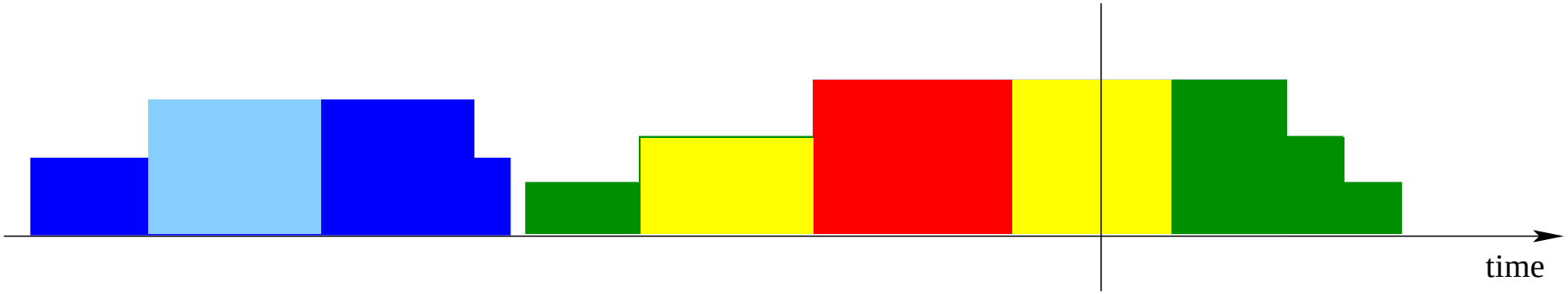
1. Job density $\delta_i = p_i/(b_i - a_i)$



2. $s(t) = \sum_{i\,:\,t \in [a_i, b_i]} \delta_i$

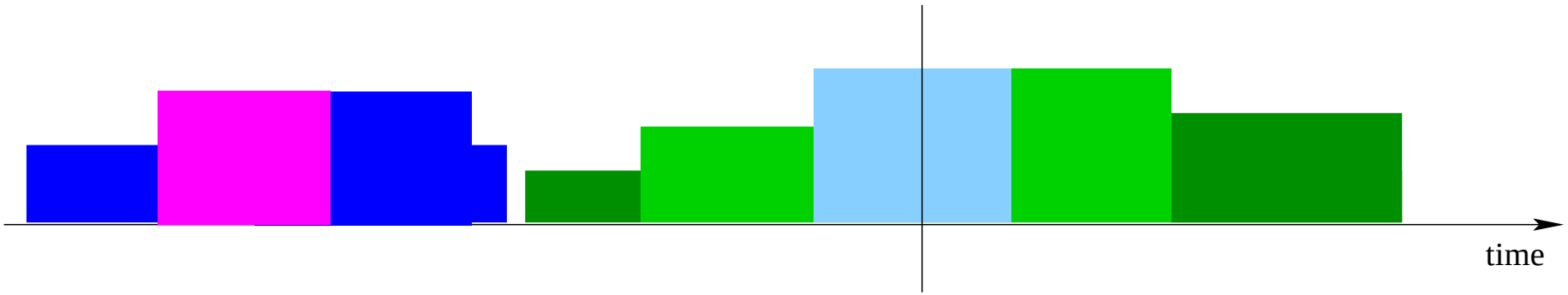


3. Use Earliest Deadline Policy



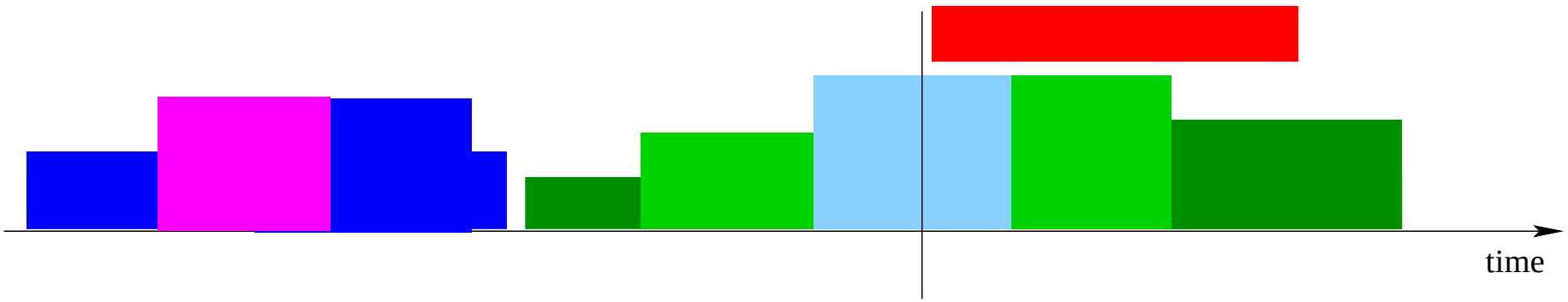
Optimal Available

At any time compute optimal schedule for remaining workload.



Optimal Available

At any time compute optimal schedule for remaining workload.

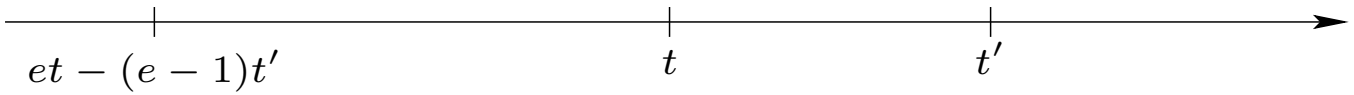


Algorithm BKP

Idea: approximate speed of optimal offline algorithm more closely

- $w(t, t_1, t_2)$ = work arrived by time t, arrival time $\geq t_1$, deadline $\leq t_2$

$$s(t) = \max_{t' > t} \frac{w(t, et - (e-1)t', t')}{(t' - t)}$$



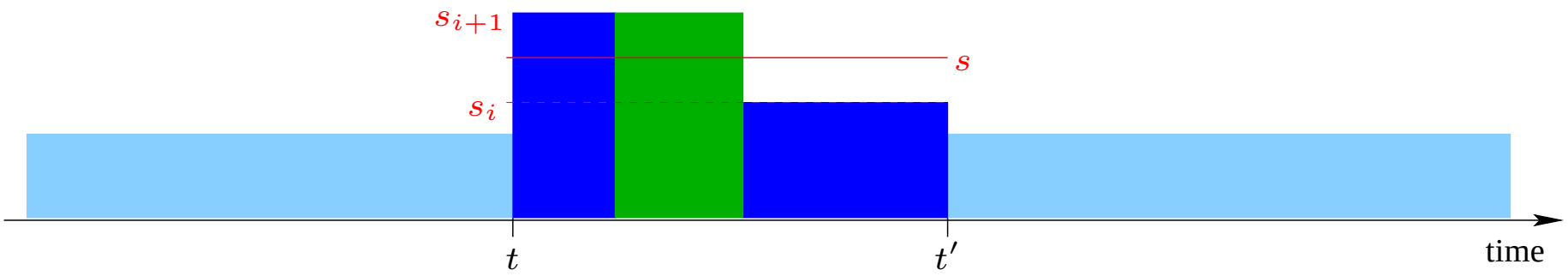
Finite speeds

Li, Yao MFCS 2005

Speed levels $s_0 < s_1 < \ldots < s_k$

Optimal offline algorithm for feasible job instances

Approximate speed in critical intervals by consecutive speed levels.



Maximum speed $s_{\max}$

Chan, Chan, Lam, Lee, Mak, Won SODA 2007

Online algorithm that is $O(1)$-competitive w.r.t. to throughput and energy

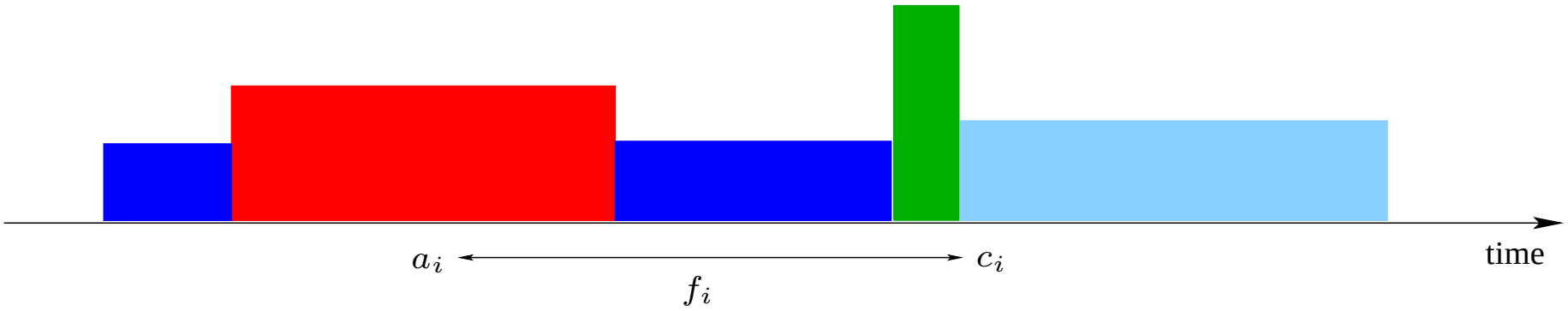
Job admission: Arrival of J with processing volume p.
Let $J_1, \ldots, J_n$ be jobs currently admitted with $b_1 \leq b_2 \leq \ldots \leq b_n$.

- J can be admitted if $\{J, J_1, \ldots, J_n\}$ full speed feasible.
- Otherwise J can be admitted if $p > 2(p_1 + \ldots + p_k)$ and $\{J, J_{k+1}, \ldots, J_n\}$ is full speed feasible, where k is the smallest integer satisfying condition.

Finished at overdue jobs are discarded.

Speed: Use that of Optimal Available if is is not larger than $s_{\max}$;
otherwise use $s_{\max}$.

Flow time minimization



- Computer systems: jobs are not labeled with deadlines
- Users expect good response times
- Flow time of J_i: $f_i = c_i - a_i$
  c_i = completion time
- Energy and flow time minimization are orthogonal objectives
  low energy $\Longrightarrow$ low speed $\Longrightarrow$ high flow times
  small flow time $\Longrightarrow$ high speed $\Longrightarrow$ high energy

Previous work

Pruhs, Uthaisombut, Woeginger 2004

- Minimize flow time given fixed energy volume V
- $p_i = 1$ for all i
- Polynomial time offline algorithm computing optimal schedules simultaneously for all V.

Our approach

Albers, Fujiwara STACS 2006

$$\min \left(\text{Energy} + \sum_{i=1}^{n} f_i \right)$$

- $\sigma = J_1, \ldots, J_n$
  J_i: a_i = arrival time
  p_i = processing volume
  preemption not allowed

- Combined objective functions for facility location, network design, TCP-acknowledgement, ...

Our results

p_i arbitrary

- Competitive ratio of $\Omega(n^{1-1/\alpha})$

$p_i = 1$

- Competitive ratio of $8e(1+\Phi)^\alpha \qquad \Phi = (1+\sqrt{5})/2 \approx 1.618$
- Offline problem polynomially solvable using dynamic programming; approach also solves problem of Pruhs, Uthaisombut, Woeginger

PhaseBalance

$S := \{$ jobs with $a_i = 0 \}$

while $S \neq \emptyset$ **do**

schedule jobs in S optimally;

$S := \{$ jobs having arrived in the meantime $\}$;

endwhile

PhaseBalance

$S := \{$ jobs with $a_i = 0 \}$

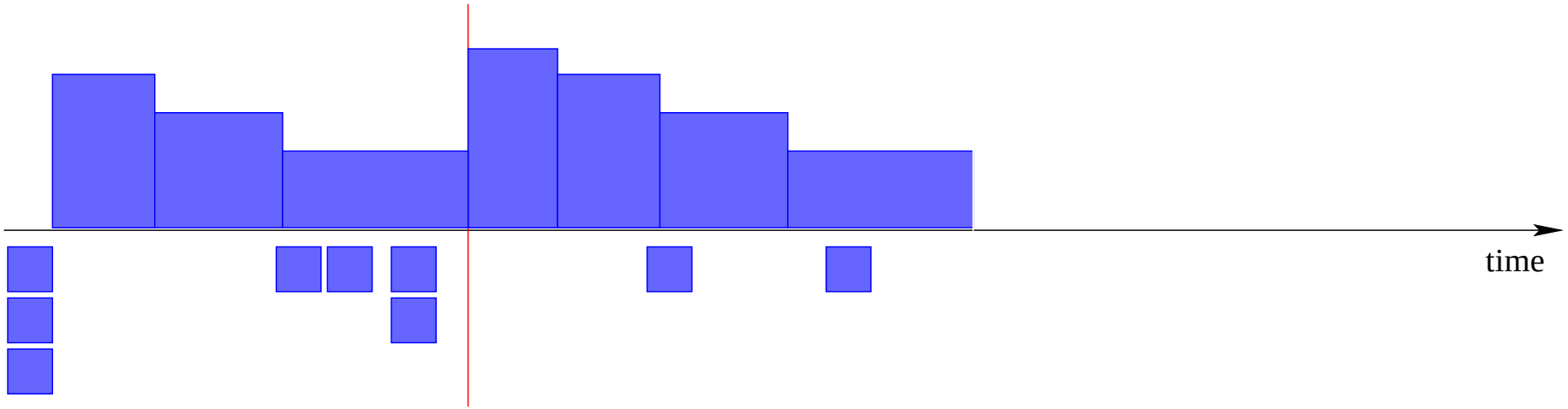
while $S \neq \emptyset$ **do**

schedule jobs in S optimally;

$S := \{$ jobs having arrived in the meantime $\}$;

endwhile

PhaseBalance



$S :=$ { jobs with $a_i = 0$ }

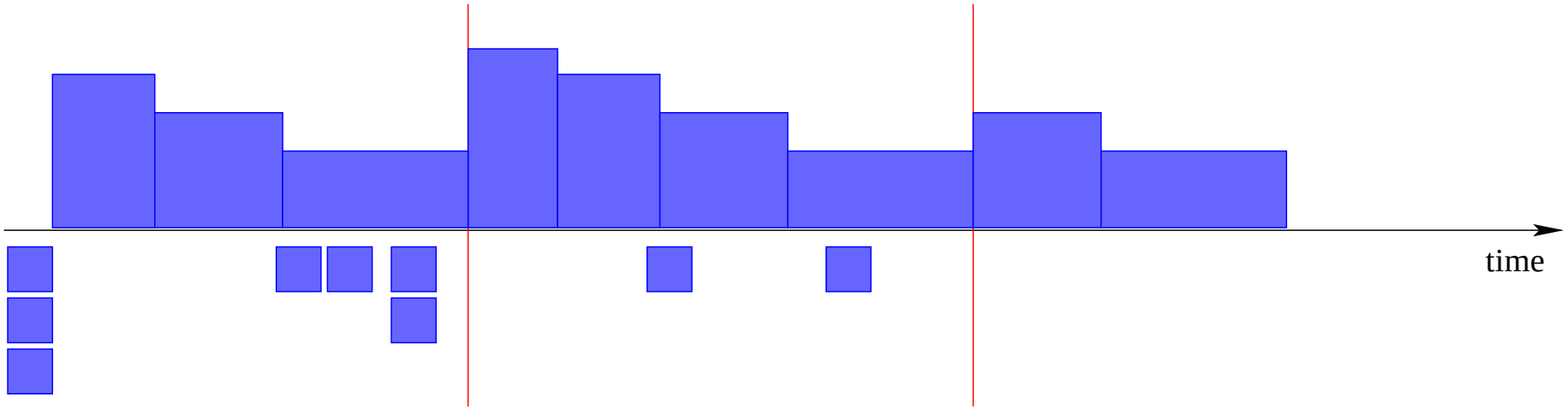
while $S \neq \emptyset$ **do**

schedule jobs in S optimally;

$S :=$ { jobs having arrived in the meantime };

endwhile

PhaseBalance



$S := \{$ jobs with $a_i = 0 \}$

while $S \neq \emptyset$ **do**

schedule jobs in S optimally;

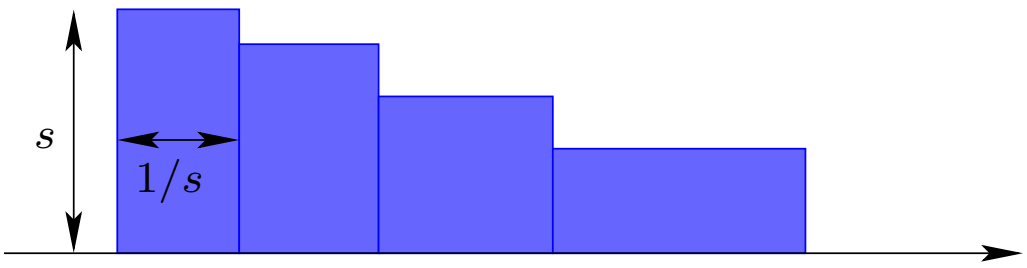
$S := \{$ jobs having arrived in the meantime $\}$;

endwhile

Phase scheduling

- n_i jobs in phase i
- Speed sequence

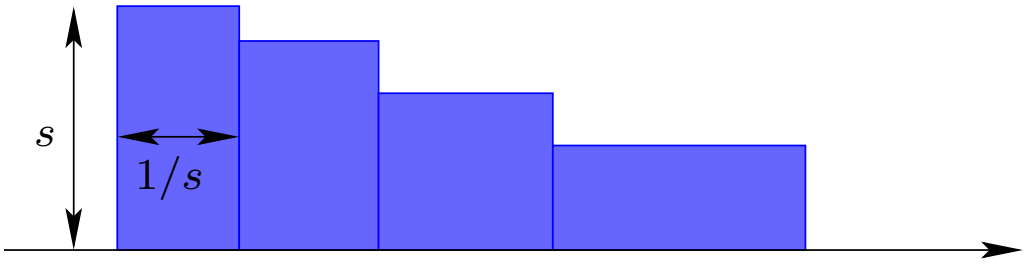
$$\sqrt[\alpha]{\frac{n_i}{\alpha-1}},\ \sqrt[\alpha]{\frac{n_i-1}{\alpha-1}},\ \ldots,\ \sqrt[\alpha]{\frac{1}{\alpha-1}}$$



Speed computation

First job of phase i

$$f(s) = s^{\alpha-1} + \frac{n_i}{s}$$



Speed computation

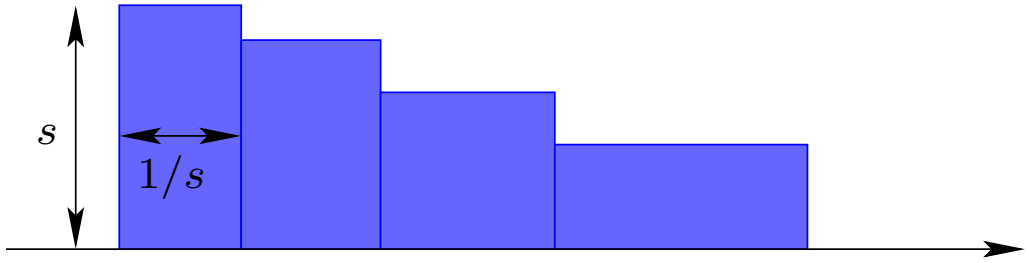
First job of phase i

$$\min f(s) = s^{\alpha-1} + \frac{n_i}{s}$$

$$f'(s) = (\alpha-1)s^{\alpha-2} - \frac{n_i}{s^2}$$

$$f'(s) = 0 \quad \Leftrightarrow \quad (\alpha-1)s^{\alpha} = n_i$$

$$\Leftrightarrow \quad s = \sqrt[\alpha]{\frac{n_i}{\alpha-1}}$$



Analysis PhaseBal

Energy in phase i

$$E(n_i) = \sum_{j=1}^{n_i} \left(\frac{n_i - j + 1}{\alpha - 1} \right)^{1-\frac{1}{\alpha}}$$

Analysis PhaseBal

Energy in phase i

$$
\begin{aligned}
E(n_i) &= \sum_{j=1}^{n_i} \left(\frac{n_i - j + 1}{\alpha - 1} \right)^{1-\frac{1}{\alpha}} \\
&\leq (\alpha - 1)^{\frac{1}{\alpha}-1} \int_1^{n_i} x^{1-\frac{1}{\alpha}} dx + (\alpha - 1)^{\frac{1}{\alpha}-1} n_i^{1-\frac{1}{\alpha}}
\end{aligned}
$$

Analysis PhaseBal

Energy in phase i

$$
\begin{aligned}
E(n_i) &= \sum_{j=1}^{n_i} \left(\frac{n_i - j + 1}{\alpha - 1} \right)^{1-\frac{1}{\alpha}} \\
&\leq (\alpha-1)^{\frac{1}{\alpha}-1} \int_1^{n_i} x^{1-\frac{1}{\alpha}} dx + (\alpha-1)^{\frac{1}{\alpha}-1} n_i^{1-\frac{1}{\alpha}} \\
&\leq (\alpha-1)^{\frac{1}{\alpha}-1} \frac{\alpha}{2\alpha-1} \; x^{2-\frac{1}{\alpha}} \Big|_1^{n_i} + (\alpha-1)^{\frac{1}{\alpha}-1} n_i^{1-\frac{1}{\alpha}}
\end{aligned}
$$

Analysis PhaseBal

Energy in phase i

$$
\begin{aligned}
E(n_i) &= \sum_{j=1}^{n_i} \left(\frac{n_i - j + 1}{\alpha - 1} \right)^{1-\frac{1}{\alpha}} \\
&\leq (\alpha-1)^{\frac{1}{\alpha}-1} \int_1^{n_i} x^{1-\frac{1}{\alpha}} dx + (\alpha-1)^{\frac{1}{\alpha}-1} n_i^{1-\frac{1}{\alpha}} \\
&\leq (\alpha-1)^{\frac{1}{\alpha}-1} \frac{\alpha}{2\alpha-1} \, x^{2-\frac{1}{\alpha}} \Big|_1^{n_i} + (\alpha-1)^{\frac{1}{\alpha}-1} n_i^{1-\frac{1}{\alpha}} \\
&\leq C_E (n_i^{2-\frac{1}{\alpha}} - 1) + (\alpha-1)^{\frac{1}{\alpha}-1} n_i^{1-\frac{1}{\alpha}}
\end{aligned}
$$

$$
C_E = (\alpha-1)^{\frac{1}{\alpha}-1} \frac{\alpha}{2\alpha-1}.
$$

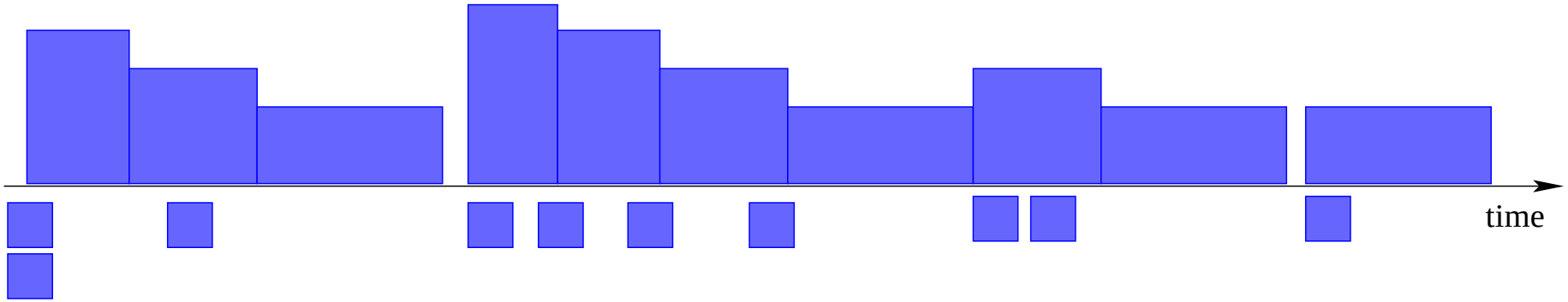
Analysis OPT

Lemma: If there are l unfinished jobs waiting

$$s \geq \sqrt[\alpha]{\frac{l}{\alpha - 1}}.$$

Lemma: Every job is finished at least as early as in the online schedule.

Offline algorithm



Sub-schedules $S_1, \ldots, S_m$.

S_j processes job with indices $j_1, \ldots, j_l$, where

$$c_i > a_{i+1} \qquad i = j_1, \ldots, j_l - 1$$

$$c_{j_l} \leq a_{j_l+1}$$

Speeds in subschedules

l jobs in interval of length T

- $s_i = \sqrt[\alpha]{\frac{l-i+1}{\alpha-1}}$ if $T \geq \sum_{i=1}^{l} 1/s_i$

- $s'_i = \sqrt[\alpha]{\frac{l-i+1+c}{\alpha-1}}$ if $T < \sum_{i=1}^{l} 1/s_i$

  c unique value with $\sum_{i=1}^{l} 1/s'_i = T$

Subproblems

$P[i, i+l]$ = subproblem with $J_i, \ldots, J_{i+l}$ if J_{i+l} must be finished by a_{i+l+1}

$C[i, i+l]$ = optimal cost of $P[i, i+l]$

Determine $C[1, n]$

Subproblems

$P[i, i+l]$ = subproblem with $J_i, \ldots, J_{i+l}$ if J_{i+l} must be finished by a_{i+l+1}

$C[i, i+l]$ = optimal cost of $P[i, i+l]$

Determine $C[1, n]$

$P[i, i+l]$: jobs processed without interruption or $P[i, k] \; + \; P[k+1, i+l]$

General setting

Bansal, Pruhs, Stein SODA 2007

$$\min \left(\text{Energy} + \sum_{i=1}^{n} w_i f_i \right)$$

p_i = arbitrary value factional weight $\overline{w_i}$ = $w_i *$ percentage to be finished

ALG:

$$s(t) = \Big(\sum_{\substack{i \\ i \text{ active at time } t}} \overline{w}_i \Big)^{1/\alpha}$$

Always schedule job with highest density w_i/p_i.

Thm: ALG is $\max\{2, 2\alpha - 2\}$-competitive.

Makespan minimization

Bunde SPAA 2006

Given energy volume V, minimize makespan.

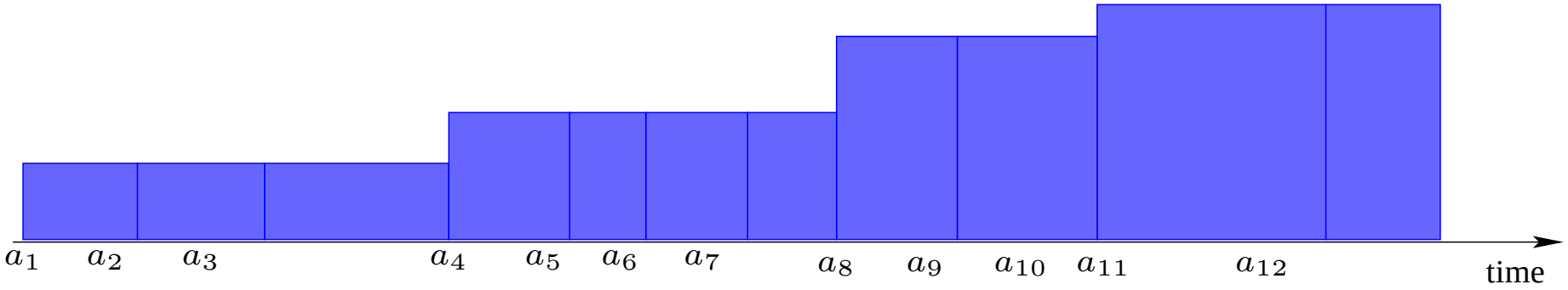
$\sigma = J_1, \ldots, J_n$

J_i: a_i = arrival time

p_i = processing volume

Optimal schedule can be computed offline in polynomial time.

Makespan minimization



Sub-schedules $S_1, \ldots, S_m$.

S_j processes job with indices $j_1, \ldots, j_l$, where

$$c_i > a_{i+1} \qquad i = j_1, \ldots, j_l - 1$$

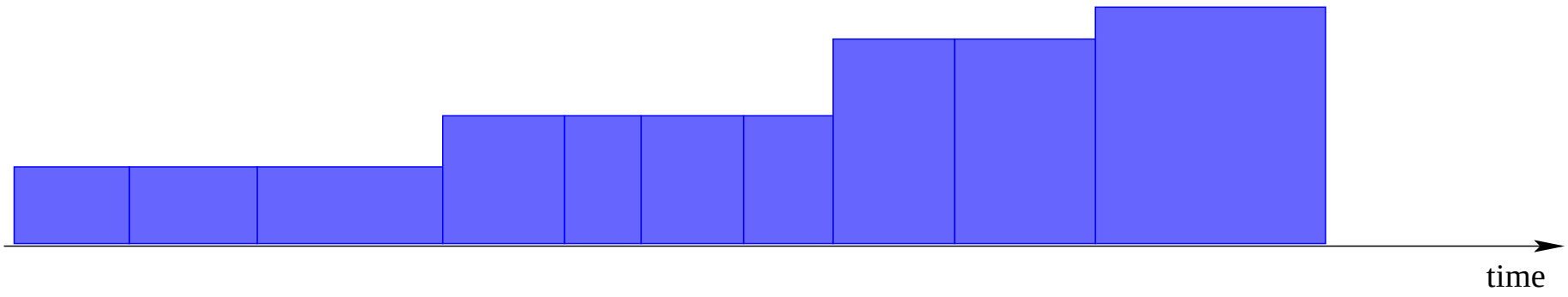
$$c_{j_l} = a_{j_l+1}$$

Properties

- Each job runs at a single speed.
- Jobs are scheduled in order or arrival time.
- Processor is not idle until the end of schedule.
- Jobs in a subschedule run at the same speed.
  jobs J_i to J_j $\quad s = \sum_{k=i}^{j} p_k/(a_{j+1} - a_i)$
- Speeds in subschedules are non-decreasing.

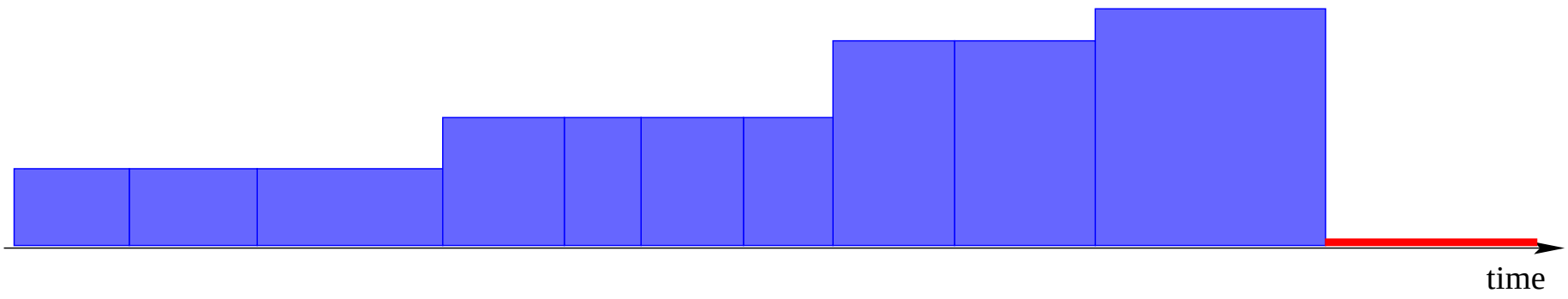
Algorithm

- Insert jobs in order of increasing arrival time into initially empty schedule.
- Each new jobs first forms a separate subschedule.
- While speeds of the last two subschedules are decreasing merge them, making use of total volume V.



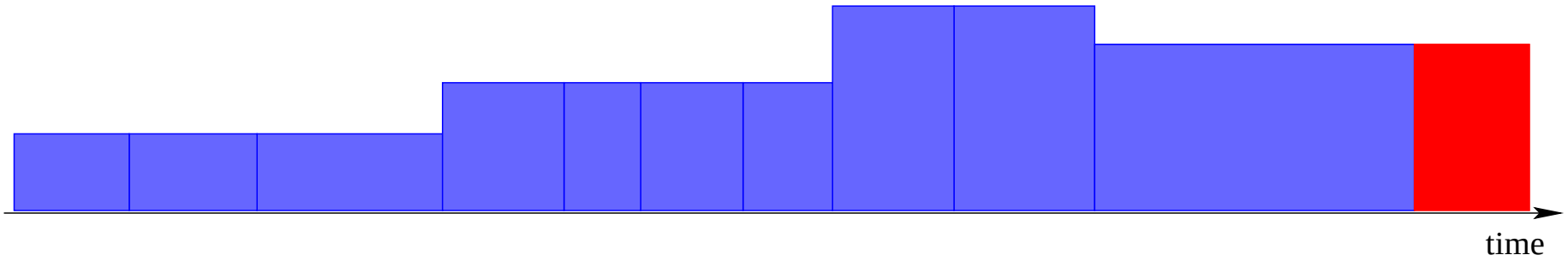
Algorithm

- Insert jobs in order increasing arrival time into initially empty schedule.
- Each new jobs first forms a separate subschedule.
- While speeds of the last two subschedules are decreasing merge them, making use of total volume V.



Algorithm

- Insert jobs in order increasing arrival time into initially empty schedule.
- Each new jobs first forms a separate subschedule.
- While speeds of the last two subschedules are decreasing merge them, making use of total volume V.



Multi-processor speed scaling

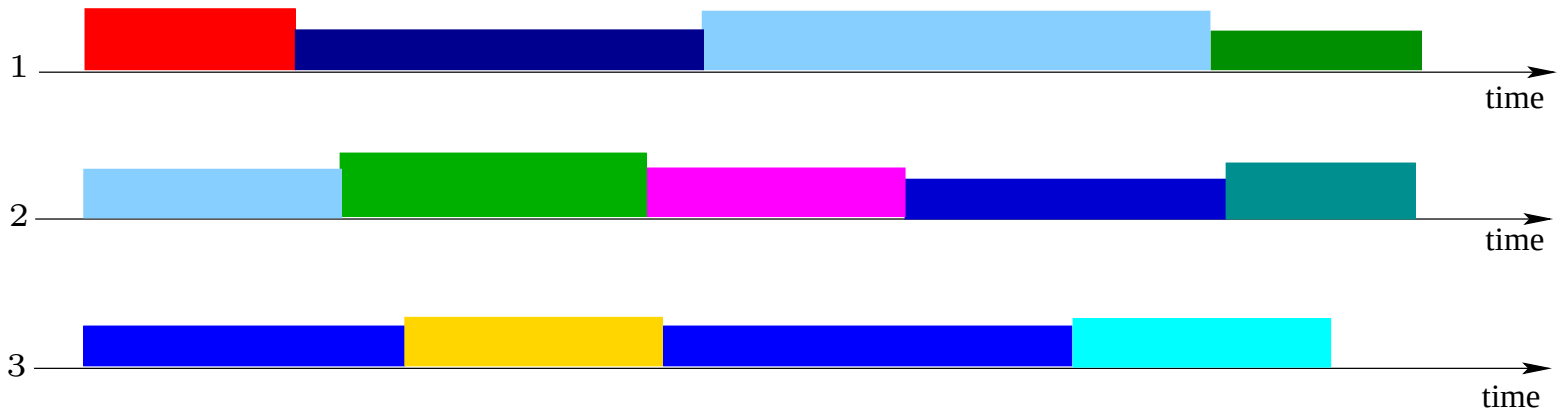
- Server systems: several CPUs
  Computer centers: power costs overtake hardware costs

- Laptops: dual-processors
  AMD ”Quad-core design”
  Architectures with 8 CPUs are being developed

- Intel: experiments with 80 CPUs on one die

Multi-processor speed scaling

Albers, Müller, Schmelzer ACM SPAA 2007



- m processors, each running at individual speed s.
- Deadline based scheduling J_i: a_i, b_i, p_i
- Preemption allowed, migration disallowed
- Construct feasible schedule minimizing total energy consumption.

Unit size jobs, $p_i = 1$

Offline setting

- Agreeable deadlines

  $a_i < a_j \implies b_i \leq b_j$

  Polynomially solvable

- Arbitrary deadlines

  NP-hard

  Approximation factor $\alpha^{\alpha} 2^{4\alpha}$

Arbitrary size jobs

Offline setting

- Common release time or common deadline
  Approximation factor $2(2 - \frac{1}{m})^{\alpha}$

- Agreeable deadlines
  Approximation factor $\alpha^{\alpha} 2^{4\alpha}$

Online setting

- $p_i = 1$, agreeable deadlines

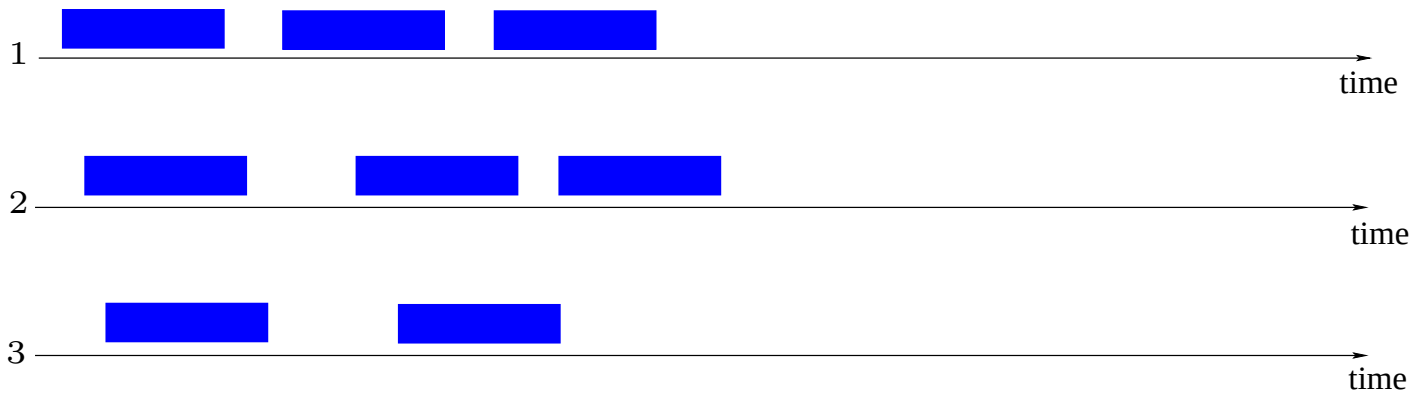
  Competitive factor $2(\frac{\alpha}{\alpha-1})^{\alpha} e^{\alpha}$

- $p_i = 1$, arbitrary deadlines

  p_i arbitrary, agreeable deadlines

  Competitive factor $\alpha^{\alpha} 2^{4\alpha}$

Unit size jobs, agreeable deadlines

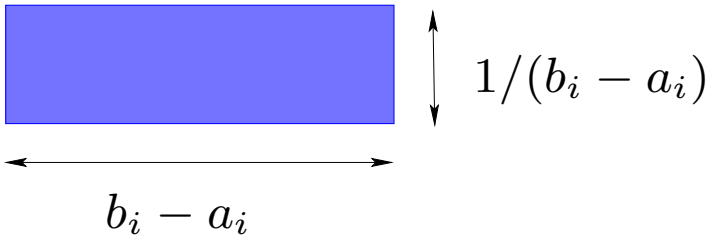


1. Sort jobs according to non-decreasing release dates.
2. Assign jobs to processors using Round Robin.
3. For each processor, compute optimal schedule.

Unit size jobs, arbitrary deadlines

1. Job density $\delta_i = 1/(b_i - a_i)$

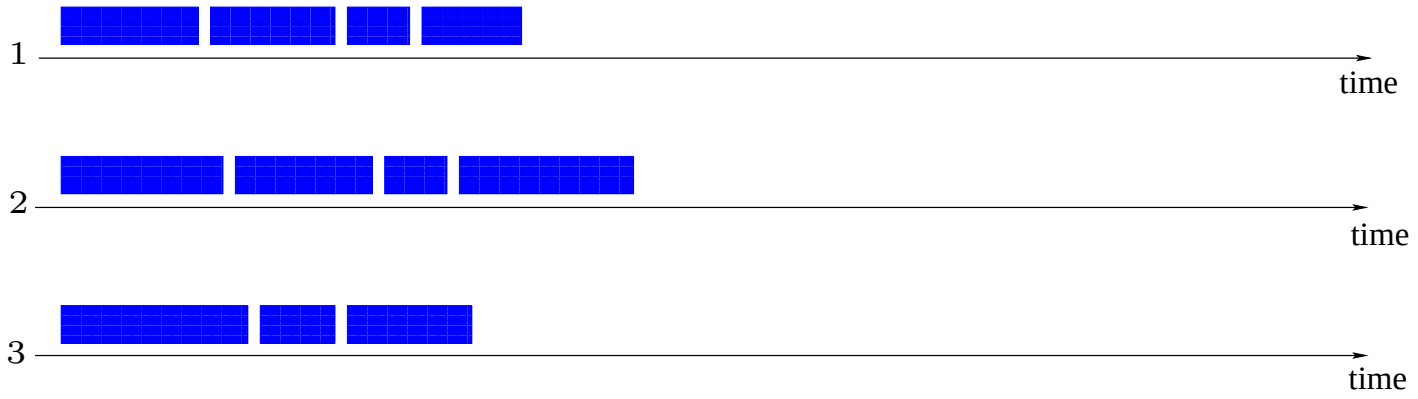
   $\Delta = \max_i \delta_i$



2. Job classes $C_k = [\Delta 2^{-k}, \Delta 2^{-k+1})$
3. Apply Round Robin to each class
4. For each processor, compute optimal schedule

Arbitrary size jobs

Common release time

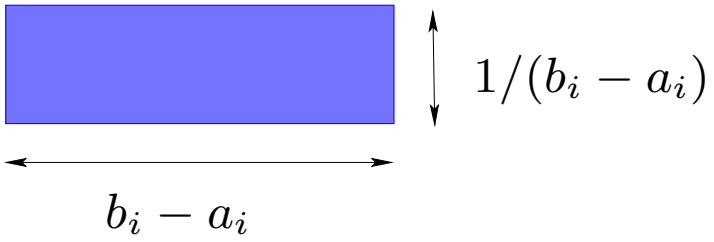


1. Sort jobs according to non-decreasing deadlines.
2. Assign jobs to processors using List scheduling.
3. For each processor, compute optimal schedule.

Arbitrary size jobs, agreeable deadlines

1. Job density $\delta_i = p_i/(b_i - a_i)$

   $\Delta = \max_i \delta_i$



2. Job classes $C_k = [\Delta 2^{-k}, \Delta 2^{-k+1})$
3. Apply Round Robin to each class
4. For each processor, compute optimal schedule

Online setting

- Unit size jobs, agreeable deadlines: Apply Round Robin; use best known online algorithm on each processor.

- Unit size jobs, arbitrary deadlines; arbitrary size jobs; agreeable deadl.:
  δ_1 = density of first job
  Open new class on arrival of job of density at most $\frac{3}{2}\delta_1 2^{-k}$
  Open new class on arrival of job of density at least $\frac{3}{2}\delta_1 2^{k}$

  Apply Round Robin for each class and optimal online algorithm for each processor.

Flow time, makespan

Bunde SPAA 2006

Minimize flow time, makespan given an energy volume V.

Job J_i: a_i, $p_i = 1$

Lemma: $\exists$ opt. solutions in which jobs are scheduled using Round Robin.

Lemma:
Flow time: The last jobs on the processors run at the same speed.
Makespan: All processors end at the same time.

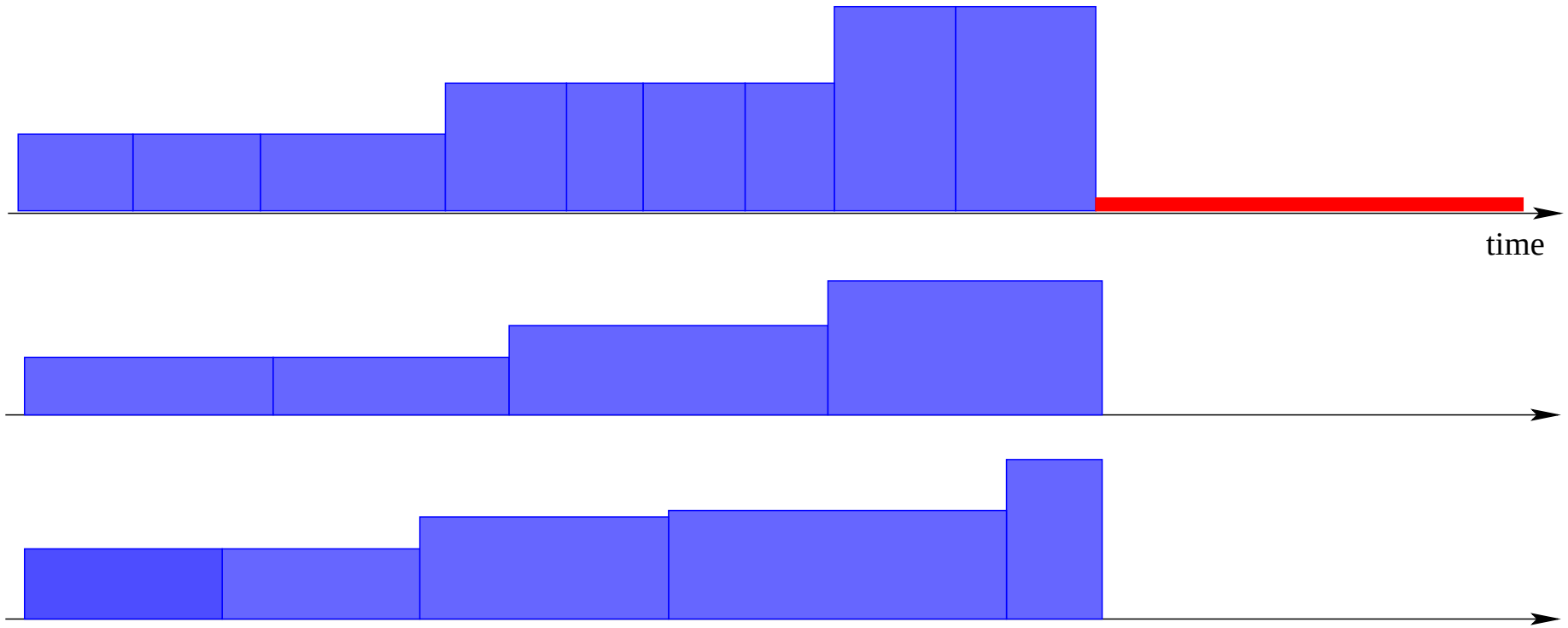
Flow time

On any processor speed of the last job determines entire schedule.

Use binary search to find optimal speeds s.t. energy consumption is V.

Makespan

While $\exists$ proc. on which last subschedule uses lower speed than previous subschedule S_j, take S_j and the last subschedules on all the processor.
Computes best speeds s.t. entire energy volume V is used.



Open problems

Single processor setting

- Bi-criteria optimization for flow and makespan.

Multi-processor setting

- Improve approximation guarantees for deadline based scenario
- Consider migration

Both settings

- Set of discrete speed levels