Dynamic Voltage Frequency Scaling (DVFS)

Prof. Dr. Jian-Jia Chen

LS 12, TU Dortmund

20, Jan., 2015

Power Consumption of CMOS Processors

- Dynamic power consumption
  - charging and discharging capacitors
- Short circuit power consumption
  - short circuit path between supply rails during switching
- Leakage
  - leaking diodes and translators
  - becomes one of the major factors due to shrinking feature sizes in semiconductor technology

Dynamic Voltage Frequency Scaling (DVFS)

- Dynamic power: $P(s)$ at speed (frequency) s
  - Adjust the supply voltage to change the execution speed
  - $s \propto \frac{(V_{dd}-V_t)^2}{V_{dd}}$, where V_{dd} is the supply voltage, and V_t is the threshold voltage
  - $P(s) \propto V_{dd}^2 s$
  - Reduced by slowing down

Details about power dissipation: Jan M. Rabaey, Anantha Chandrakasan, and Borivoje Nikolic, "Digital Integrated Circuits," Second Edition, Prentice-Hall, ISBN-10: 9780130909961 [Chapters 5.5, 5.6, 6.2, 11.7]

Today's Focus

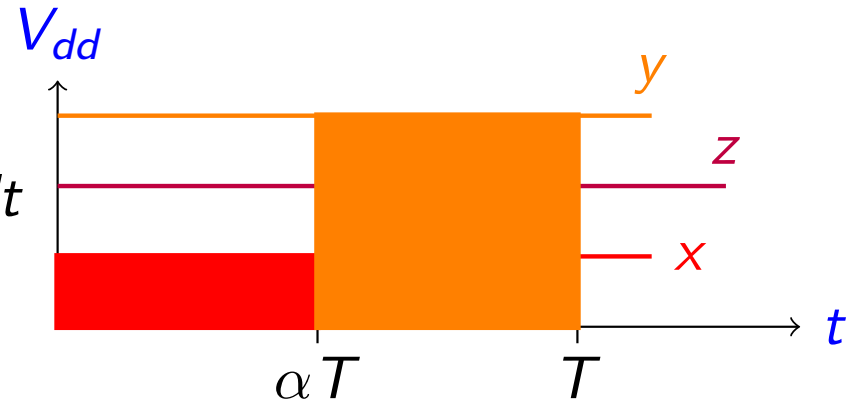
How to minimize the energy consumption by applying DVFS?

- Simplified DVFS power model for a given speed s
  - Suppose that $P(s)$ is simply a convex and increasing function
  - Suppose that $P(s) = s^{\gamma}$ for $\gamma > 1$
- Simplified execution model
  - The execution time is proportional to $\frac{1}{s}$.

Energy Consumption with Different Speeds

Execute task in fixed time T with variable voltage $V_{dd}(t)$ or variable speed $s(t)$:

- Delay: $\propto \frac{1}{s(t)}$
- Energy: $\propto \int_0^T V_{dd}^2(t)s(t)dt$



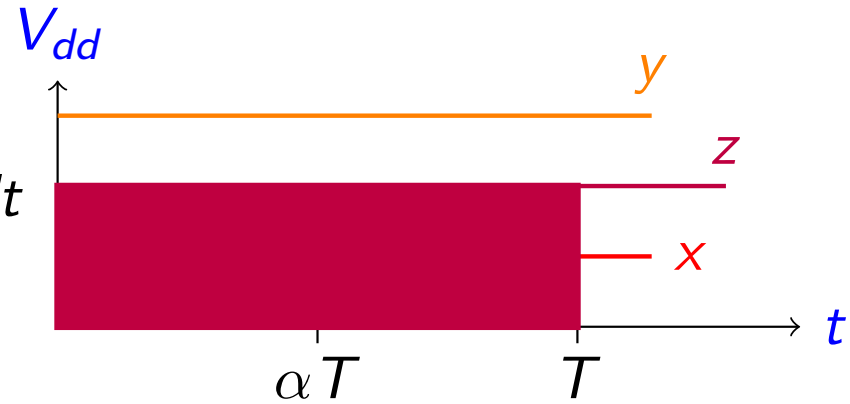
Consider two cases:

- Case A: Execute at voltage (speed) x for αT time units and voltage (speed) y for $(1-\alpha)T$ time units with $0 \leq \alpha \leq 1$:
  - Energy: $T \cdot (\alpha P(x) + (1-\alpha)P(y))$
- Case B: Execute at voltage (speed) z for T time units, in which $\min\{x,y\} \leq z \leq \max\{x,y\}$ and $z = \alpha \cdot x + (1-\alpha) \cdot y$:
  - Energy: $T \cdot P(z)$

Energy Consumption with Different Speeds

Execute task in fixed time T with variable voltage $V_{dd}(t)$ or variable speed $s(t)$:

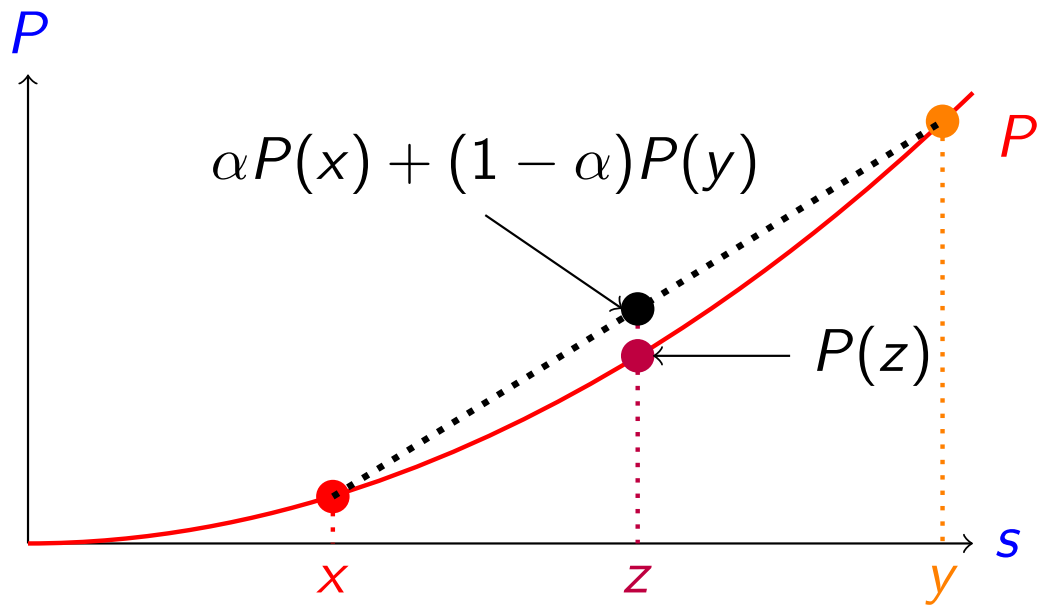
- Delay: $\propto \frac{1}{s(t)}$
- Energy: $\propto \int_0^T V_{dd}^2(t)s(t)dt$



Consider two cases:

- Case A: Execute at voltage (speed) x for αT time units and voltage (speed) y for $(1-\alpha)T$ time units with $0 \leq \alpha \leq 1$:
  - Energy: $T \cdot (\alpha P(x) + (1-\alpha)P(y))$
- Case B: Execute at voltage (speed) z for T time units, in which $\min\{x,y\} \leq z \leq \max\{x,y\}$ and $z = \alpha \cdot x + (1-\alpha) \cdot y$:
  - Energy: $T \cdot P(z)$

Convex Function



$$z = \alpha \cdot x + (1-\alpha) \cdot y$$

- If possible, running at a constant speed (voltage) minimizes the energy consumption for dynamic voltage scaling:
  - case A is always worse if the power consumption is a convex function of the supply voltage

Outline

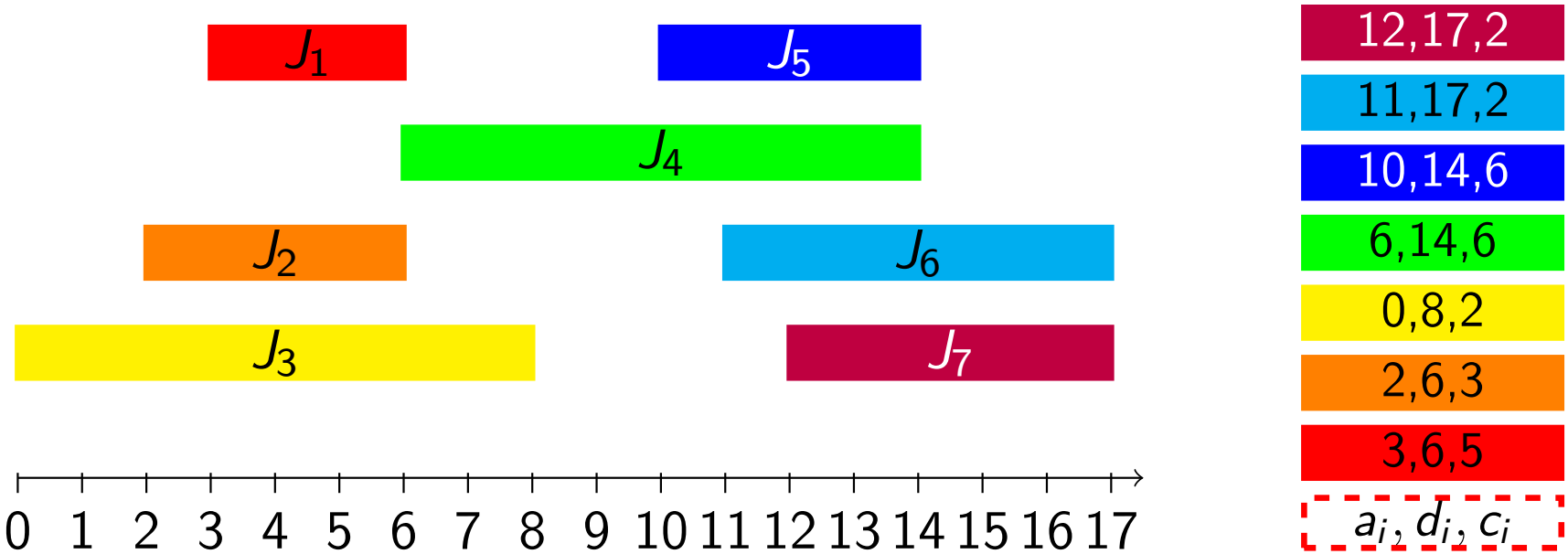
YDS Algorithm

Discrete Speeds

Problem Definition

- Input:
  - A set $\mathcal{J}$ of n jobs, in which job $J_j \in \mathcal{J}$ is with
    - arrival time a_j
    - absolute deadline d_j
    - execution time c_j at speed 1
  - A platform with DVFS capability with speed in $[0, \infty)$ and power consumption $P(s)$ is a convex function
- Output:
  - A feasible schedule to meet the timing constraints, which minimizes the energy consumption.
  - Again, we will use EDF, as it is optimal for deadline satisfactions.

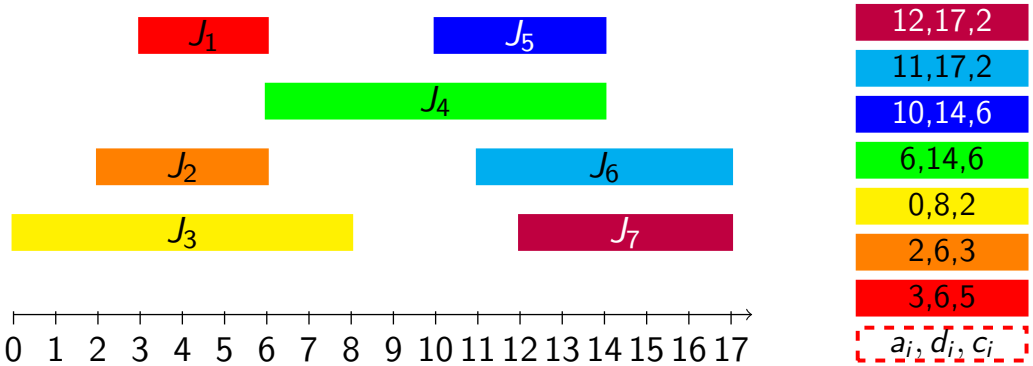
Intensity



Intensity $G([z, z'])$ in some time interval $[z, z']$: average accumulated execution time of all jobs that have arrival and deadline in $[z, z']$:

$$G([z, z']) = \sum_{a_i \geq z \text{ and } d_i \leq z'} c_i/(z' - z)$$

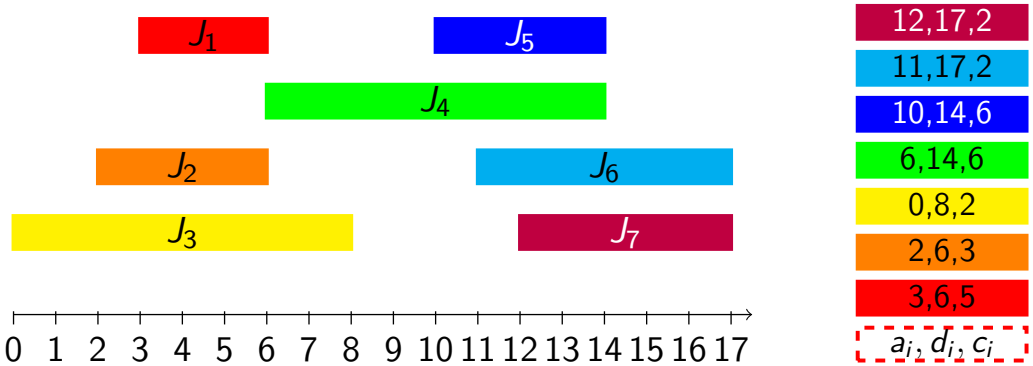
YDS Algorithm: Step 1



Step1: Execute jobs in the interval with the highest intensity by using the earliest-deadline first schedule and running at the intensity as the speed.

$G([0,6]) = (5+3)/6 = 8/6,$
$G([0,8]) = (5+3+2)/(8-0) = 10/8,$
$G([0,14]) = (5+3+2+6+6)/14 = 11/7,$
$G([0,17]) = 26/17,$
$G([2,6]) = (5+3)/(6-2) = 2,$
$G([2,14]) = (5+3+6+6)/(14-2) = 5/3,$
$G([2,17]) = 26/15,$
$G([3,6]) = 5/3$
$G([3,14]) = (5+6+6)/11 = 17/11,$
$G([3,17]) = (5+6+6+2+2)/14 = 21/14,$
$G([6,14]) = 12/8 = 12/8,$
$G([6,17]) = (6+6+2+2)/11 = 16/11,$
$G([10,14]) = 6/4,$
$G([10,17]) = 10/7,$
$G([11,17]) = 4/6,$
$G([12,17]) = 2/5$

YDS Algorithm: Step 1

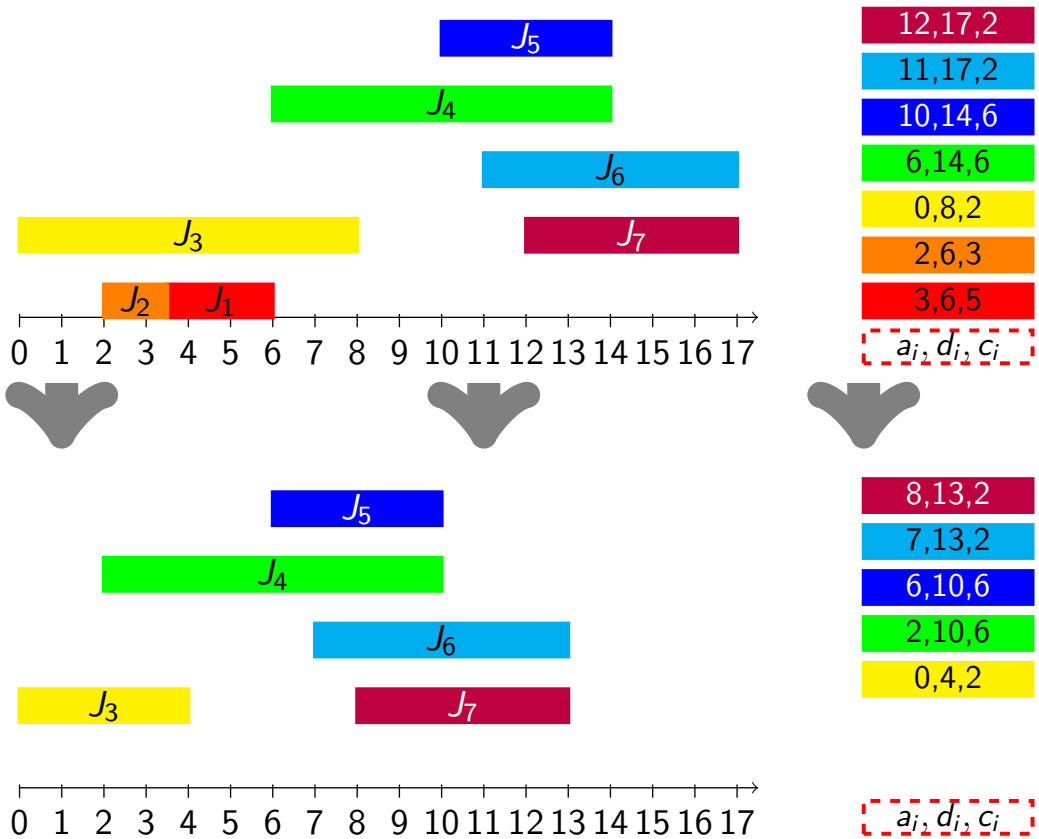


Step1: Execute jobs in the interval with the highest intensity by using the earliest-deadline first schedule and running at the intensity as the speed.

$G([0,6]) = (5+3)/6 = 8/6,$
$G([0,8]) = (5+3+2)/(8-0) = 10/8,$
$G([0,14]) = (5+3+2+6+6)/14 = 11/7,$
$G([0,17]) = 26/17,$
$G([2,6]) = (5+3)/(6-2) = 2,$
$G([2,14]) = (5+3+6+6)/(14-2) = 5/3,$
$G([2,17]) = 26/15,$
$G([3,6]) = 5/3$
$G([3,14]) = (5+6+6)/11 = 17/11,$
$G([3,17]) = (5+6+6+2+2)/14 = 21/14,$
$G([6,14]) = 12/8 = 12/8,$
$G([6,17]) = (6+6+2+2)/11 = 16/11,$
$G([10,14]) = 6/4,$
$G([10,17]) = 10/7,$
$G([11,17]) = 4/6,$
$G([12,17]) = 2/5$

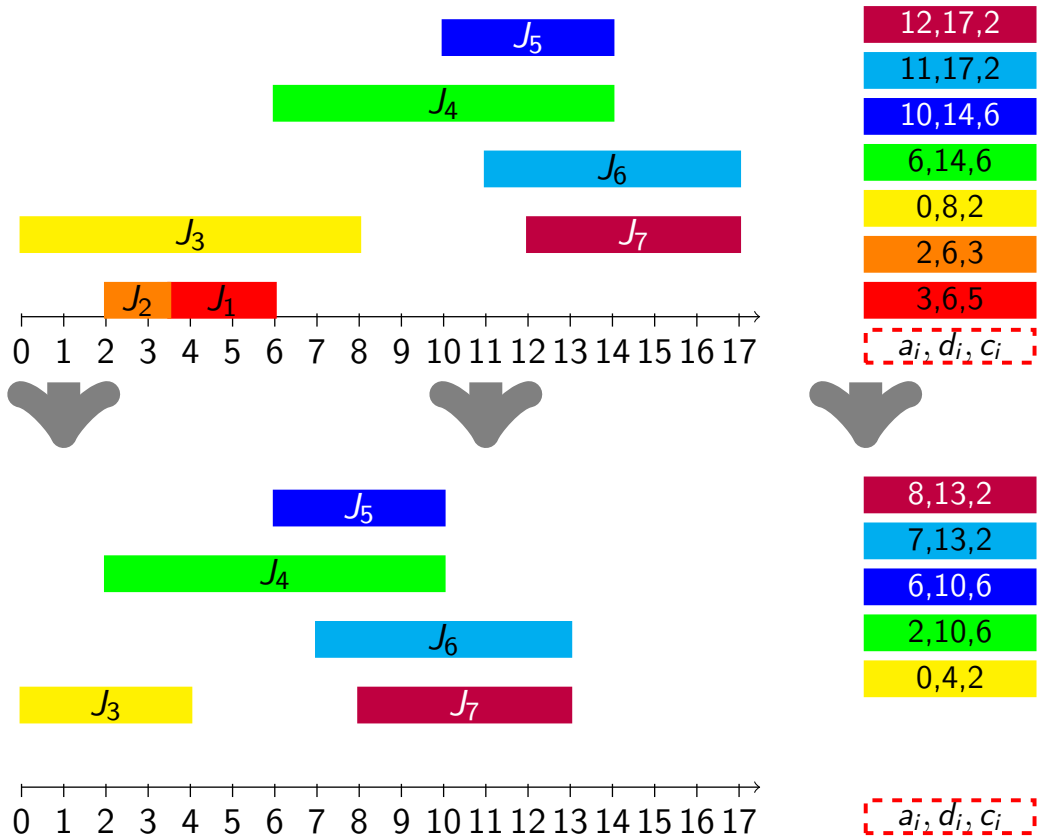
YDS Algorithm: Step 2

J_1 and J_2 run at speed 2 from time 2 to time 6.



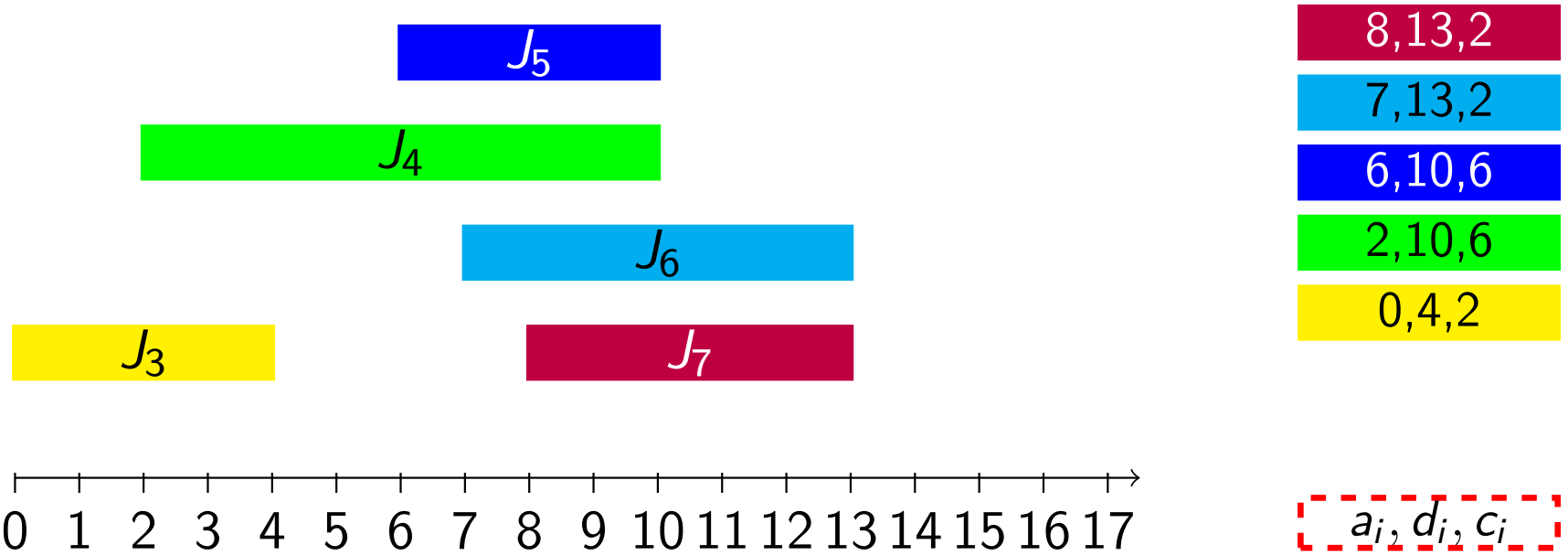
YDS Algorithm: Step 2

Step 2: Adjust the arrival times and deadlines by excluding the possibility to execute at the previous critical intervals.



YDS Algorithm: Step 3

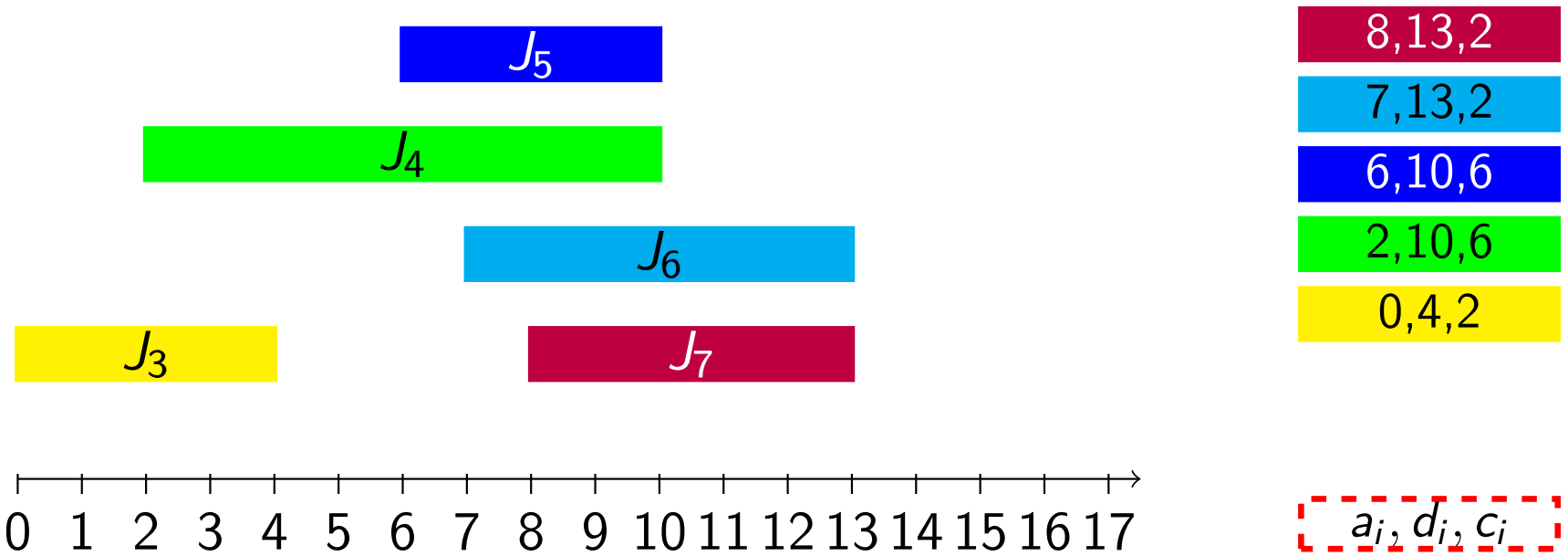
Step 3: Run the algorithm for the revised input again



G([0,4])=2/4, G([0,10]) = 14/10, G([0,13])=18/13, G([2,10])=12/8,
G([2,13]) = 16/11, G([6,10])=6/4, G([7,13])=4/6, G([8,13])=4/5

YDS Algorithm: Step 3

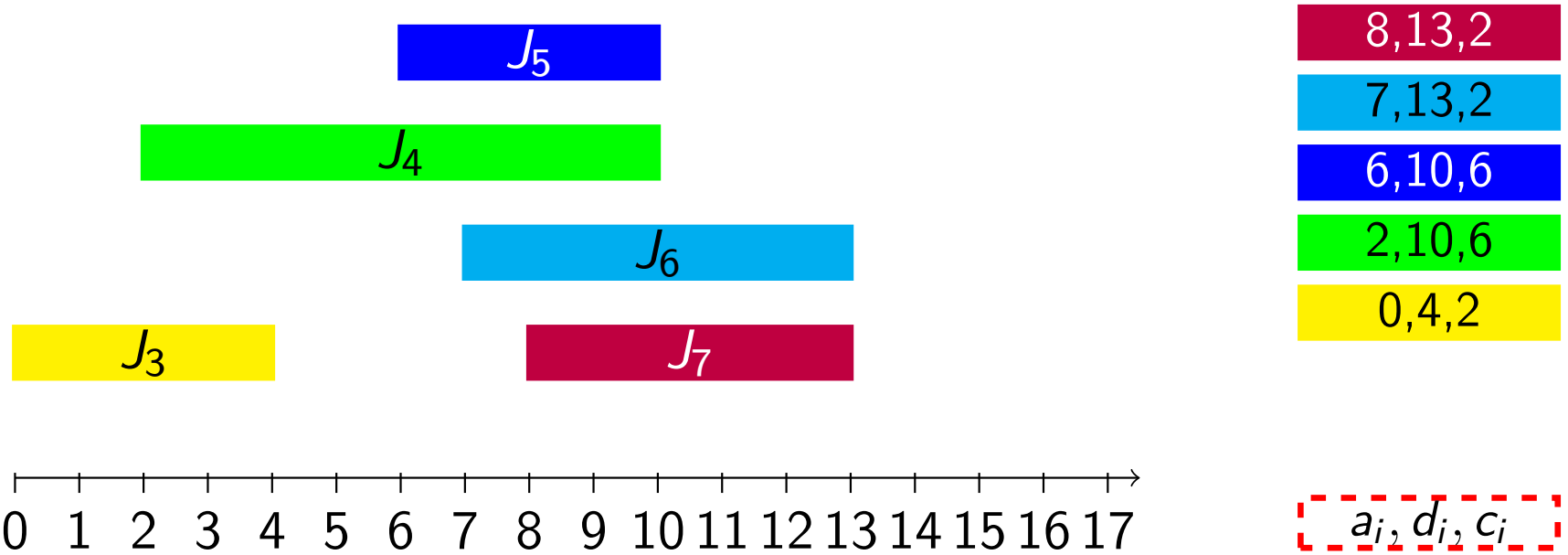
Step 3: Run the algorithm for the revised input again



G([0,4])=2/4, G([0,10]) = 14/10, G([0,13])=18/13, G([2,10])=12/8,
G([2,13]) = 16/11, G([6,10])=6/4, G([7,13])=4/6, G([8,13])=4/5

YDS Algorithm: Step 3

Step 3: Run the algorithm for the revised input again



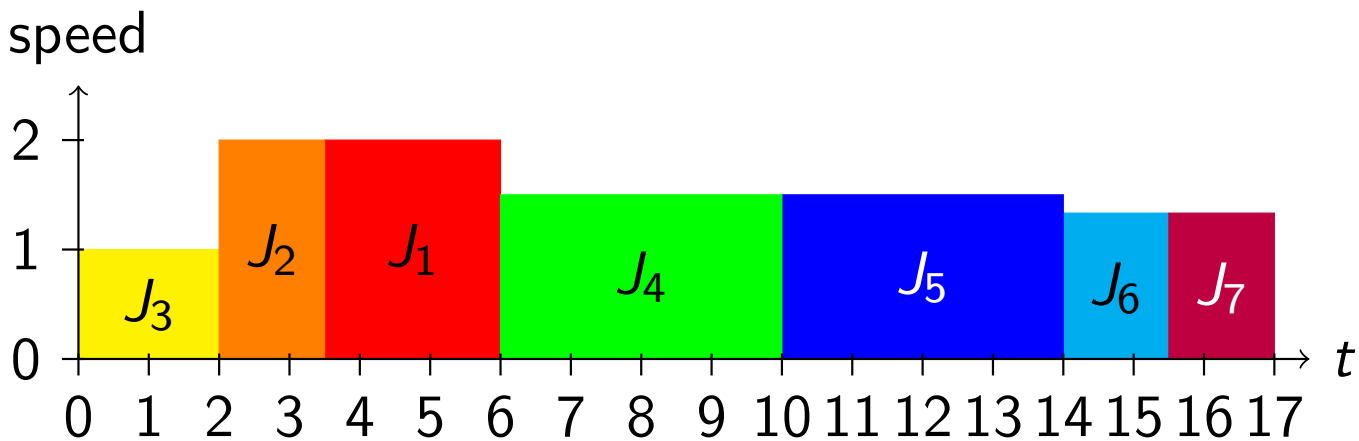
G([0,4])=2/4, G([0,10]) = 14/10, G([0,13])=18/13, G([2,10])=12/8,
G([2,13]) = 16/11, G([6,10])=6/4, G([7,13])=4/6, G([8,13])=4/5

J_4 and J_5 run at speed 1.5.

YDS Algorithm

- Step 3: Run the algorithm for the revised input again
  - After the previous steps (time interval 2 to 10 is trimmed)

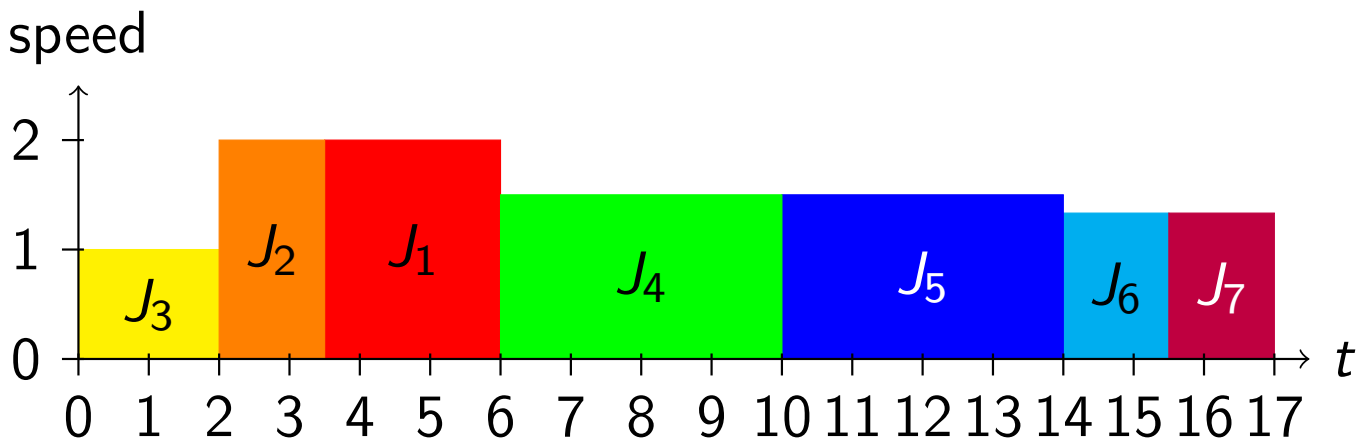
J_3		J_5		J_7	
orig	new	orig	new	orig	new
$(0, 4, 2)$		$(7, 13, 12)$		$(8, 13, 12)$	



YDS Algorithm

- Step 3: Run the algorithm for the revised input again
  - After the previous steps (time interval 2 to 10 is trimmed)

J_3		J_5		J_7	
orig	new	orig	new	orig	new
$(0, 4, 2)$	$(0, 2, 2)$	$(7, 13, 12)$	$(2, 5, 2)$	$(8, 13, 12)$	$(2, 5, 2)$

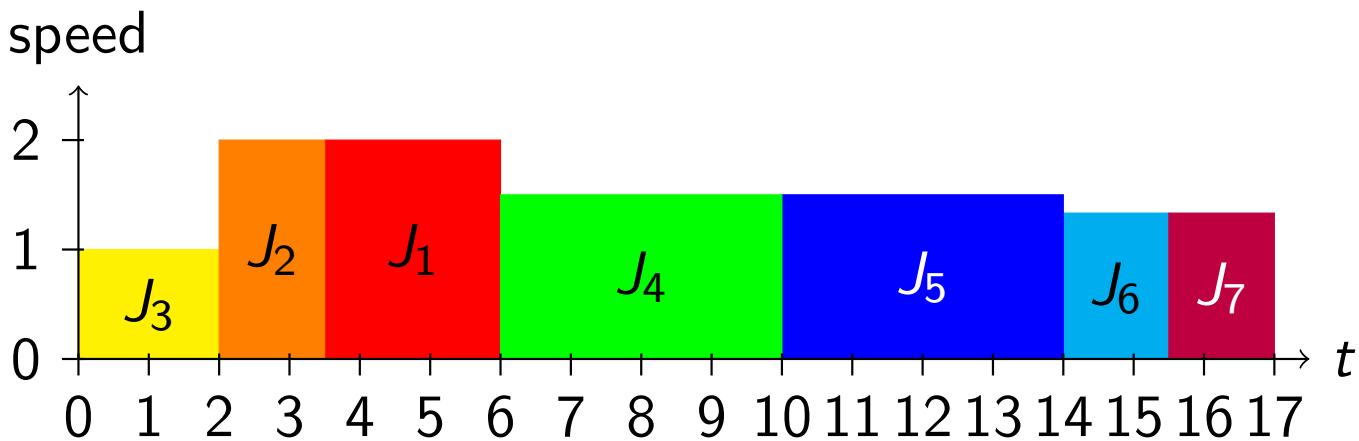


YDS Algorithm

- Step 3: Run the algorithm for the revised input again
  - After the previous steps (time interval 2 to 10 is trimmed)

J_3		J_5		J_7	
orig	new	orig	new	orig	new
$(0, 4, 2)$	$(0, 2, 2)$	$(7, 13, 12)$	$(2, 5, 2)$	$(8, 13, 12)$	$(2, 5, 2)$

  - J_6 and J_7 are executed at speed $\frac{4}{3}$, and then J_1 is executed at speed 1.



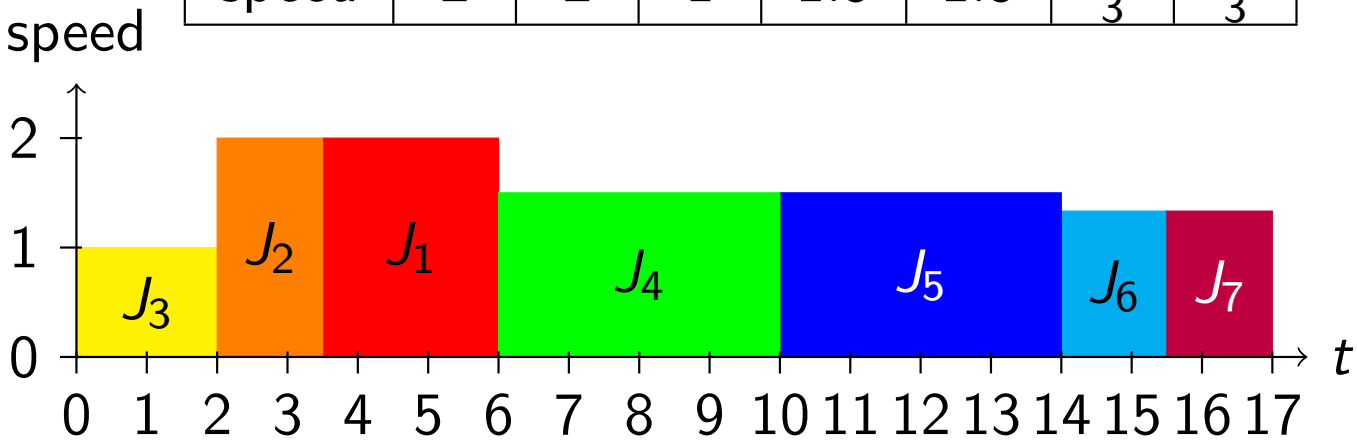
YDS Algorithm

- Step 3: Run the algorithm for the revised input again
  - After the previous steps (time interval 2 to 10 is trimmed)

J_3		J_5		J_7	
orig	new	orig	new	orig	new
$(0, 4, 2)$	$(0, 2, 2)$	$(7, 13, 12)$	$(2, 5, 2)$	$(8, 13, 12)$	$(2, 5, 2)$

  - J_6 and J_7 are executed at speed $\frac{4}{3}$, and then J_1 is executed at speed 1.
- Step 4: Put pieces together by using EDF

	J_1	J_2	J_3	J_4	J_5	J_6	J_7
speed	2	2	1	1.5	1.5	$\frac{4}{3}$	$\frac{4}{3}$



Remarks of YDS Algorithm

Th algorithm guarantees the minimum energy consumption while satisfying the timing constraints

- The time complexity is $O(|\mathcal{J}|^3)$
  - Finding the critical interval can be done in $O(|\mathcal{J}|^2)$
  - There are $O(|\mathcal{J}|)$ iterations

Optimality of YDS Algorithm

The problem can be formulated into a convex programming

- Objective is a convex function for energy consumption
- Constraints are linear constraints for execution time satisfactions.

Details are in Nikhil Bansal, Kirk Pruhs: Speed Scaling to Manage Temperature. STACS 2005: 460-471 (Page 469-470 for the detailed proof.)

Outline

Reality of DVFS

DVS technology

- Intel's SpeedStep - 2 speeds
- AMD's PowerNow - 9 speeds
- Intel's Foxton technology - 64 speeds

Applying YDS Algorithm

- Deriving the speeds first
- Use linear combinations of speeds to run tasks
- The time complexity is $O(|\mathcal{J}|^3)$

Reality of DVFS

DVS technology

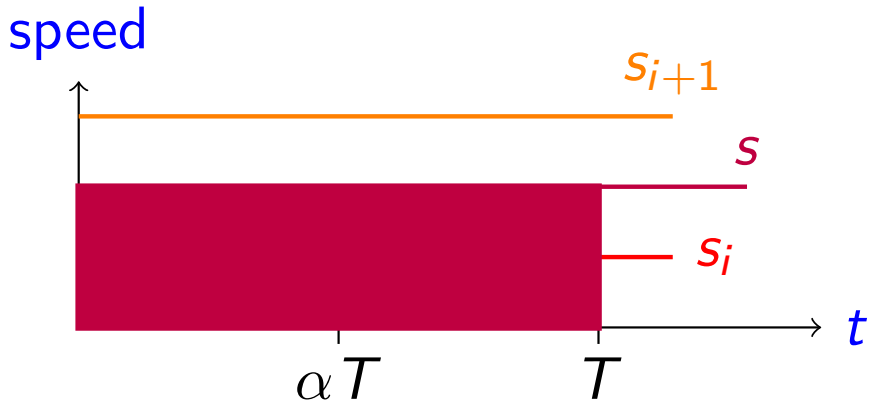
- Intel's SpeedStep - 2 speeds
- AMD's PowerNow - 9 speeds
- Intel's Foxton technology - 64 speeds

Applying YDS Algorithm

- Deriving the speeds first
- Use linear combinations of speeds to run tasks
- The time complexity is $O(|\mathcal{J}|^3)$

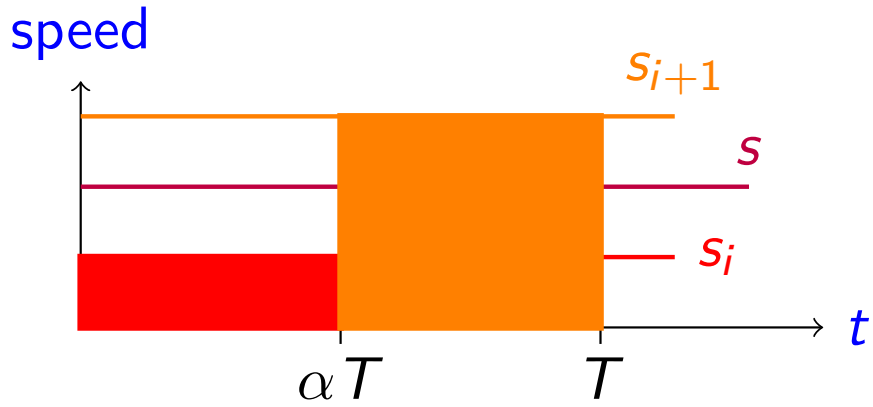
Our objective here: reduce the time complexity to $O(K|\mathcal{J}|\log|\mathcal{J}|)$ for systems with K discrete speeds.

Linear Combination of Speeds



- The optimal solution is to execute at speed s for T time units

Linear Combination of Speeds

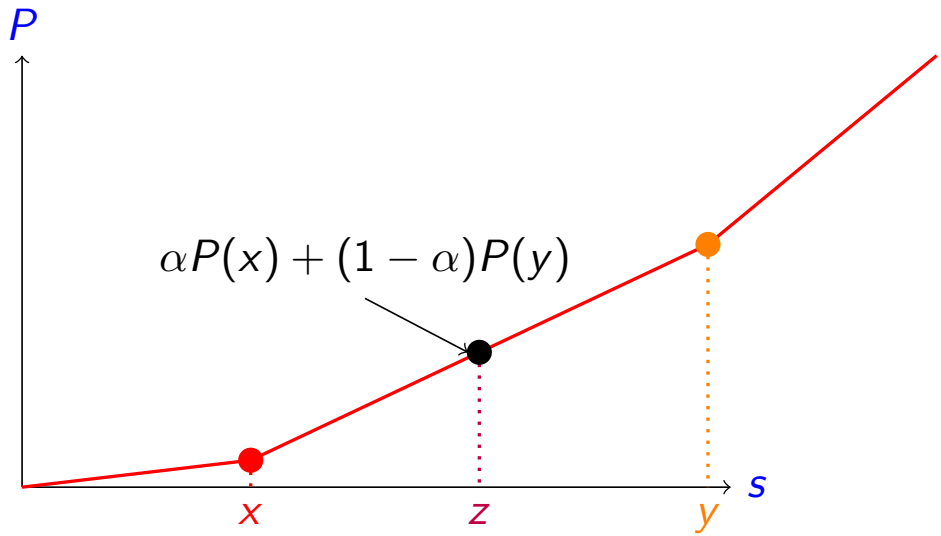


- The optimal solution is to execute at speed s for T time units
- The optimal solution for systems with discrete speeds is to execute at two closest speeds $s_i < s \leq s_{i+1}$ such that

$$s_i \alpha T + s_{i+1}(1-\alpha)T = sT \Rightarrow \alpha = \frac{s_{i+1} - s}{s_{i+1} - s_i}.$$

  - Run at speed s_i for $\frac{s_{i+1}-s}{s_{i+1}-s_i}$ time units and at speed s_{i+1} for $\frac{s-s_i}{s_{i+1}-s_i}$ time units.

Linear Combination of Speeds



$$z = \alpha \cdot x + (1-\alpha) \cdot y$$

Assume that the power consumption is convex and increasing.

- Using two available adjacent speeds (x, y) to "emulate" speed z minimizes the energy consumption.
- Exercise: Please verify how to convert YDS Algorithm from continuous speeds to discrete speeds.