Power and Energy Issues for Real-Time Systems

Prof. Dr. Jian-Jia Chen

LS 12, TU Dortmund

13, Jul., 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

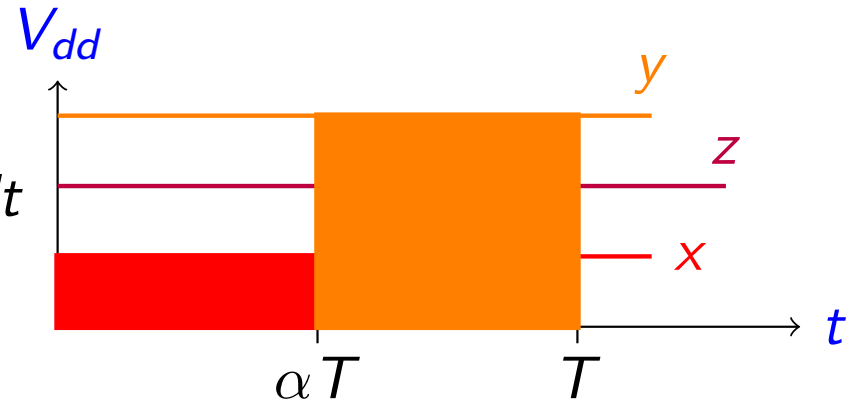
How to minimize the energy consumption by applying DVFS?

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



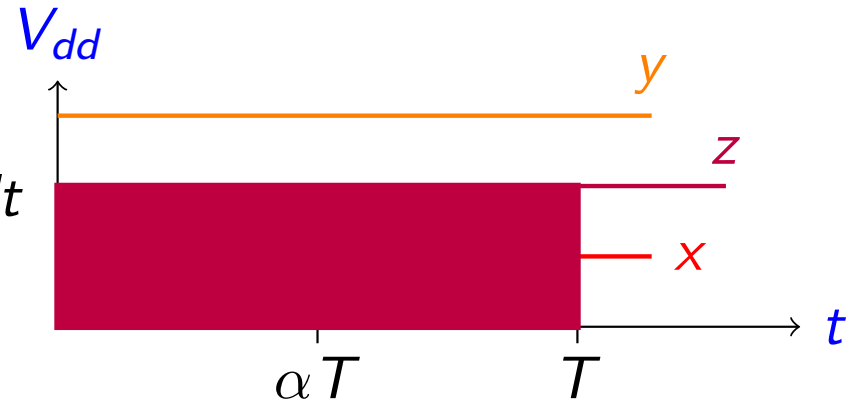
Consider two cases:

- 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 $s(t)$:

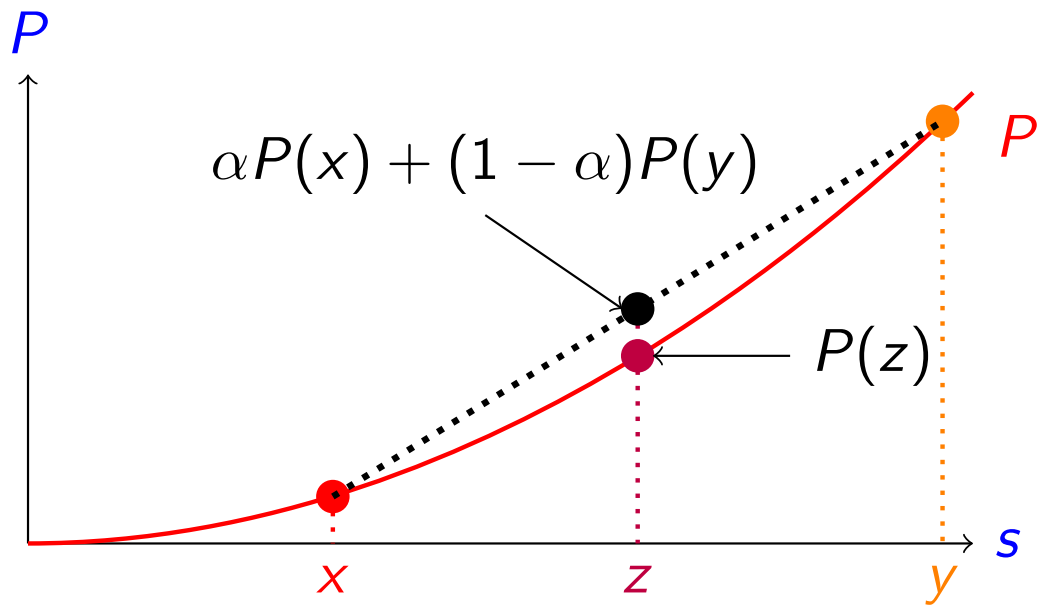
- Delay: $\propto \frac{1}{s(t)}$
- Energy: $\propto \int_0^T V_{dd}^2(t)s(t)dt$



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

Energy and Power Optimization for Multiprocessor Systems

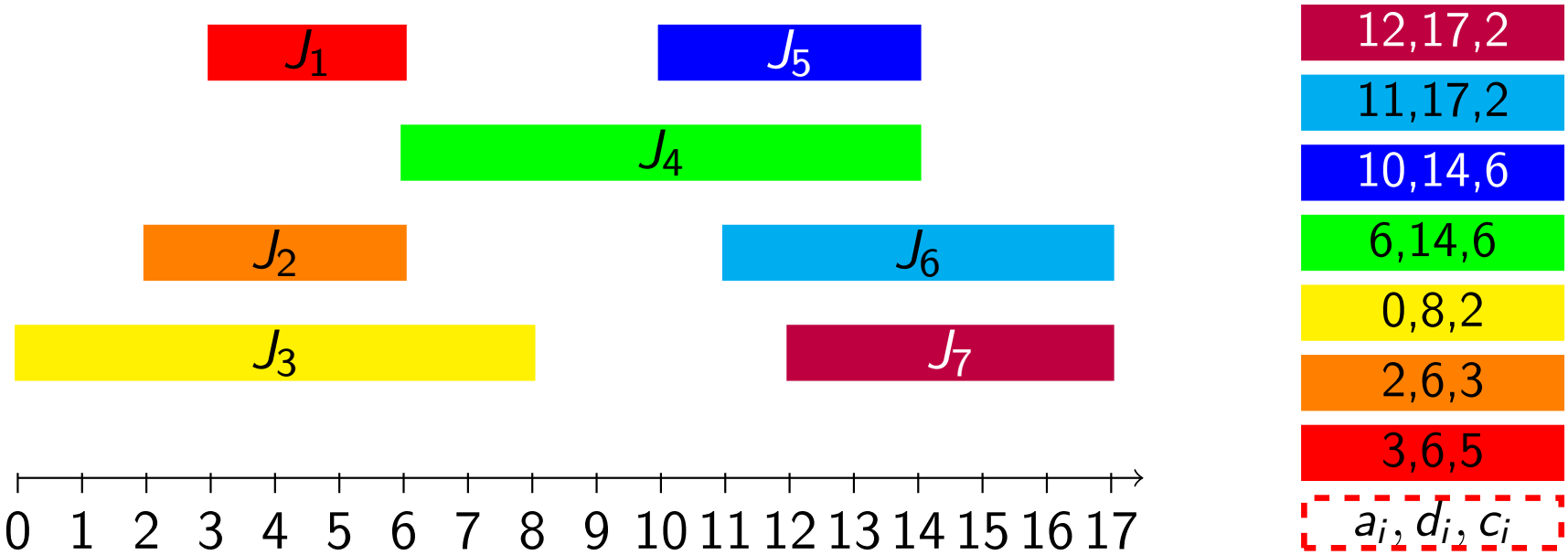
Appendix for SFA

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.

F. Frances Yao, Alan J. Demers, Scott Shenker: A Scheduling Model for Reduced CPU Energy. FOCS 1995: 374-382

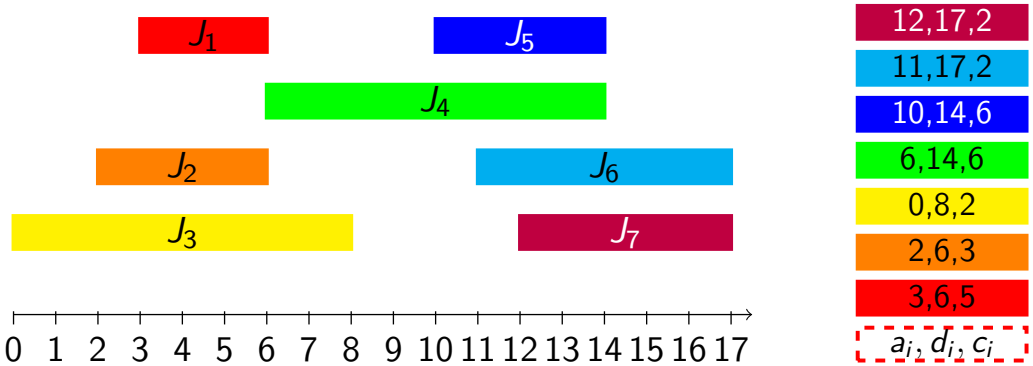
Intensity



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

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

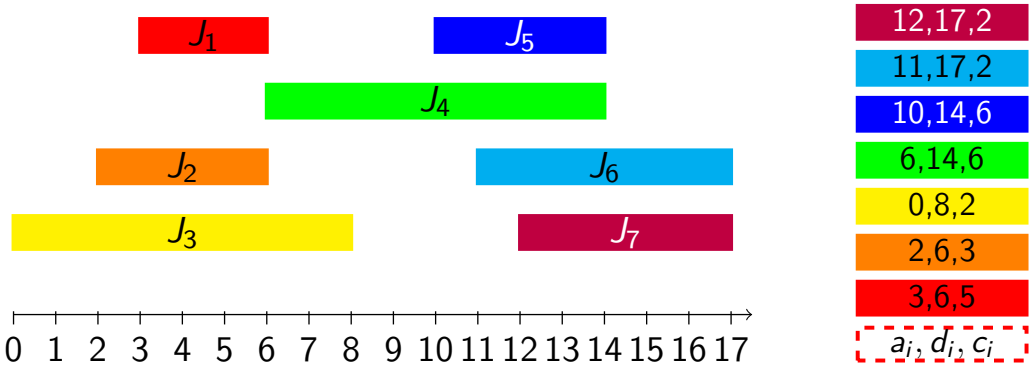
YDS Algorithm: Step 1



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

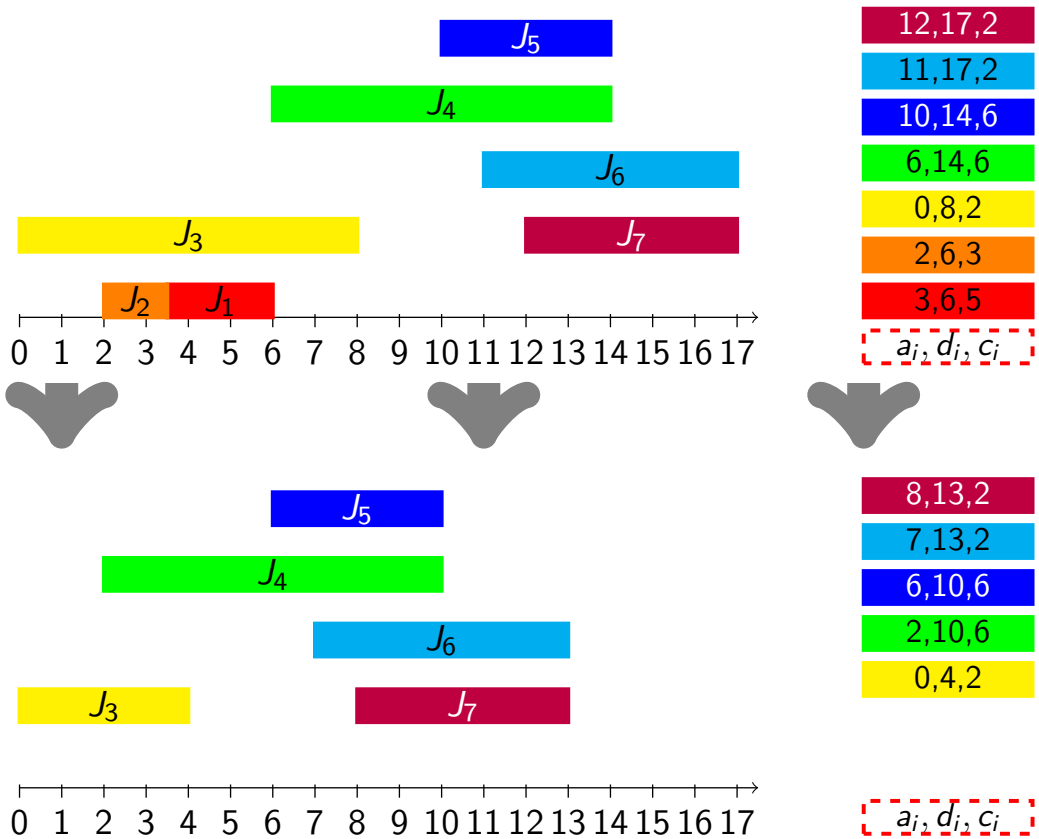


Step1: Execute jobs in the interval with the highest intensity by using the earliest-deadline first schedule and running at the intensity as the speed.

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

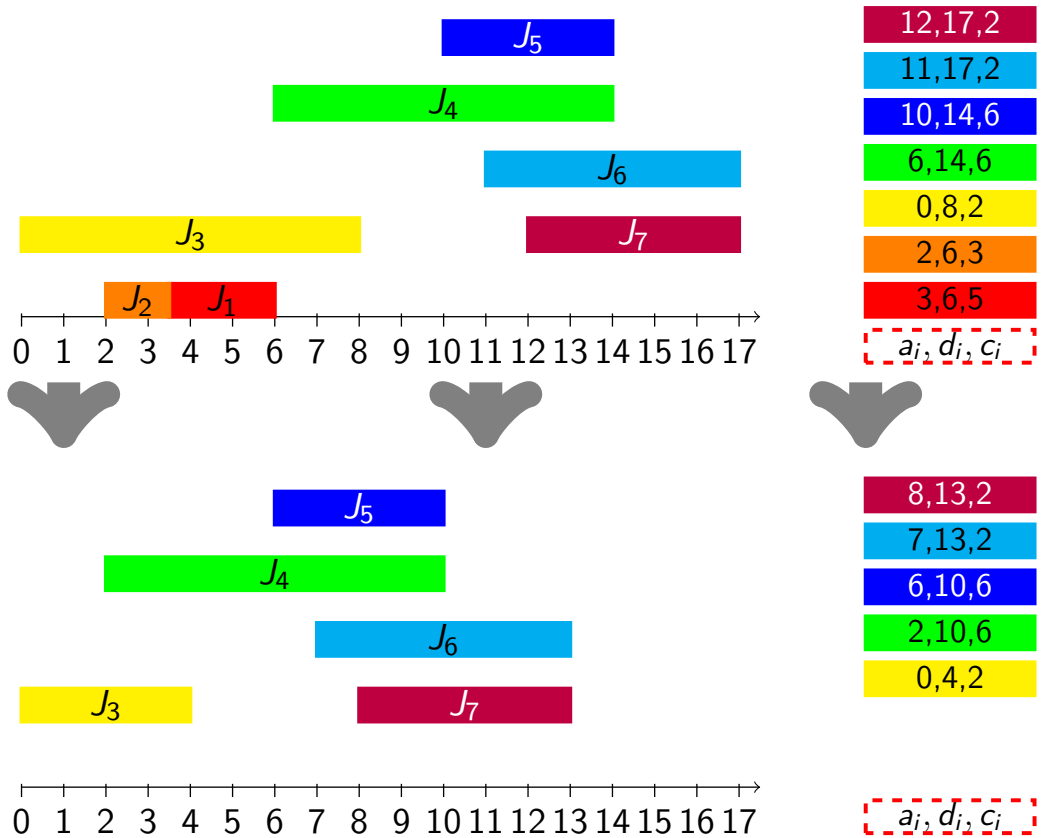
YDS Algorithm: Step 2

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



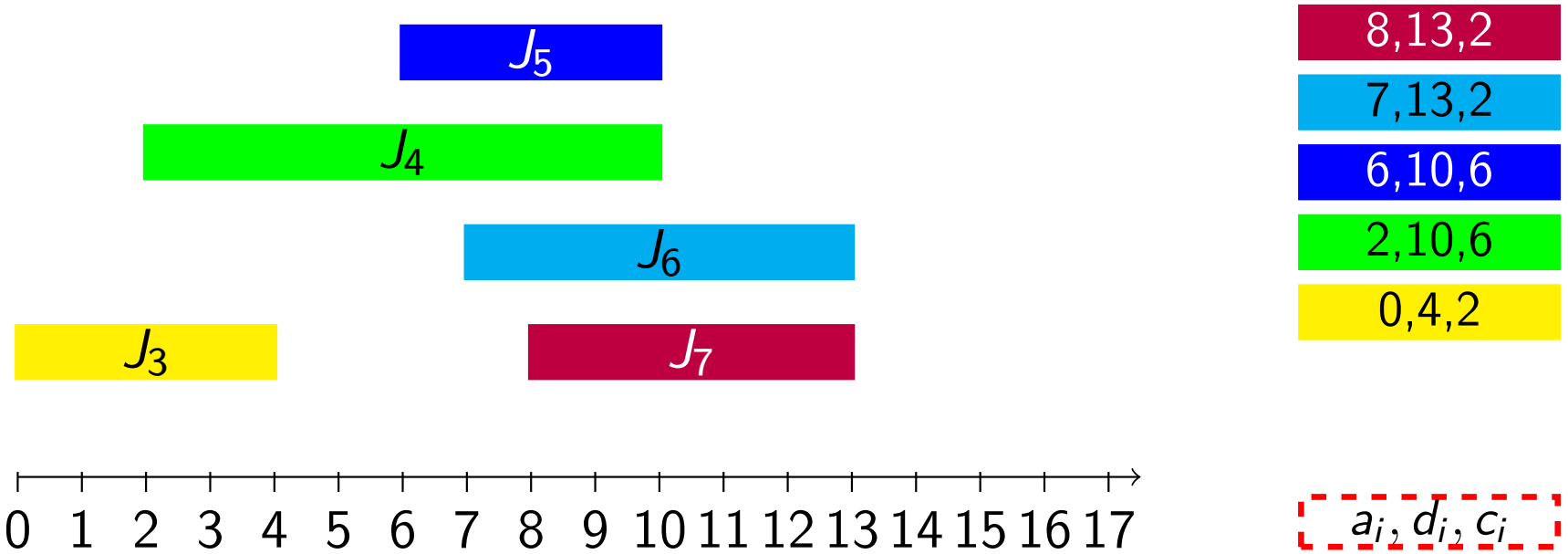
YDS Algorithm: Step 2

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



YDS Algorithm: Step 3

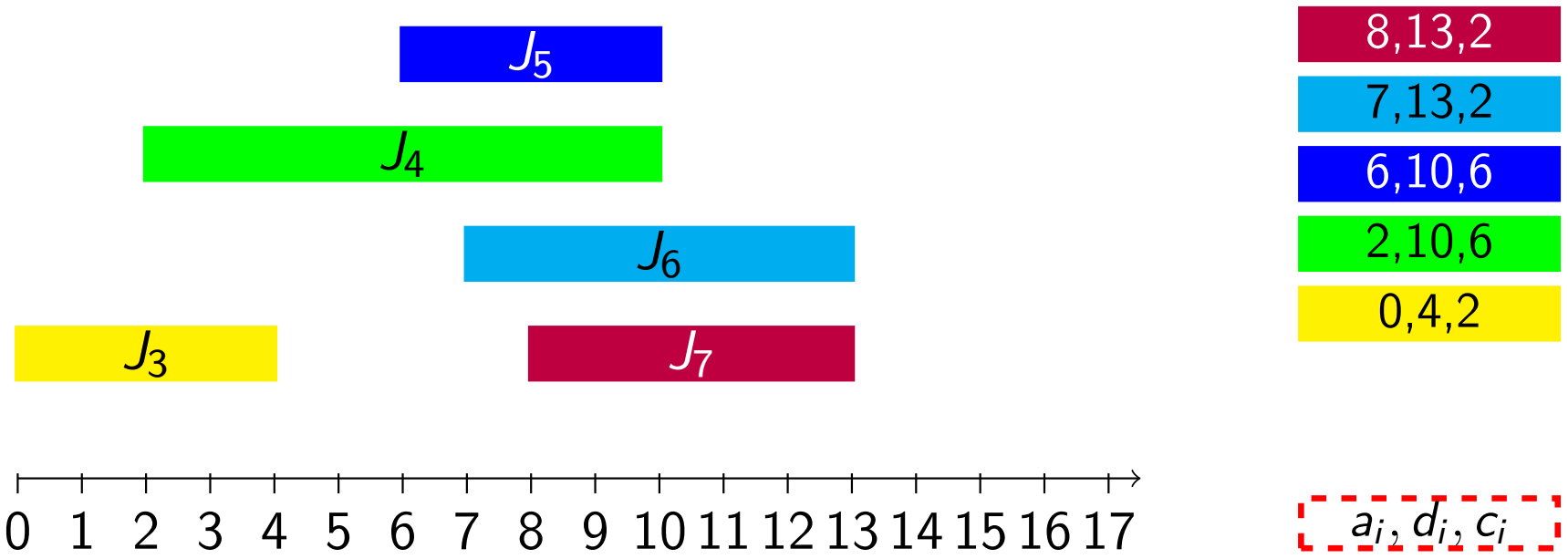
Step 3: Run the algorithm for the revised input again



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

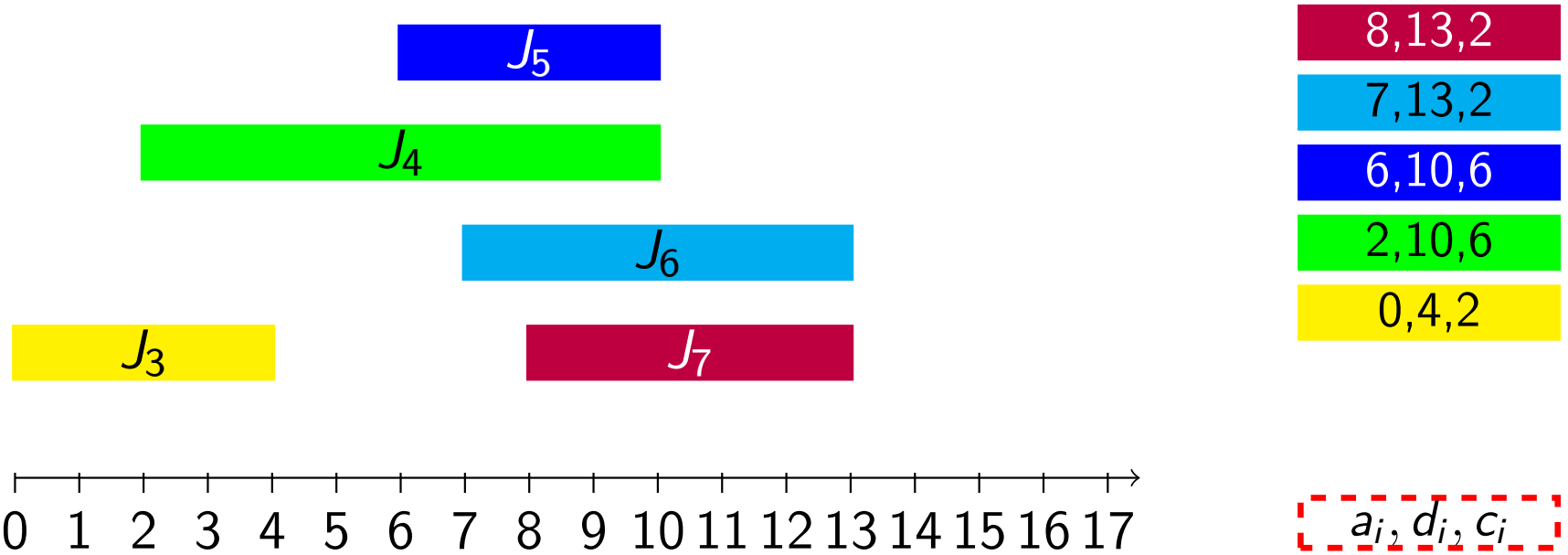
Step 3: Run the algorithm for the revised input again



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



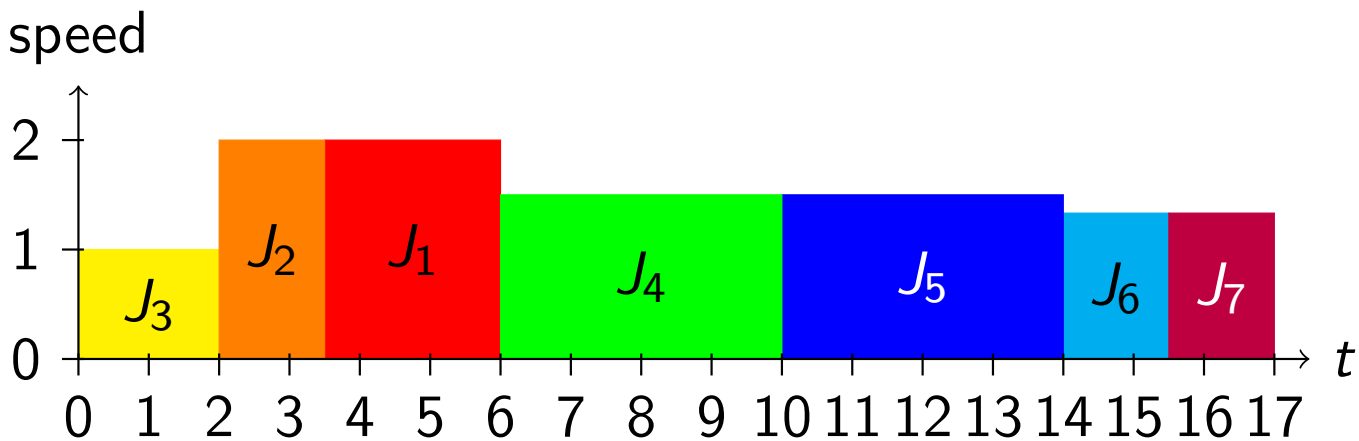
G([0,4])=2/4, G([0,10]) = 14/10, G([0,13])=18/13, G([2,10])=12/8,
G([2,13]) = 16/11, G([6,10])=6/4, G([7,13])=4/6, G([8,13])=4/5

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)

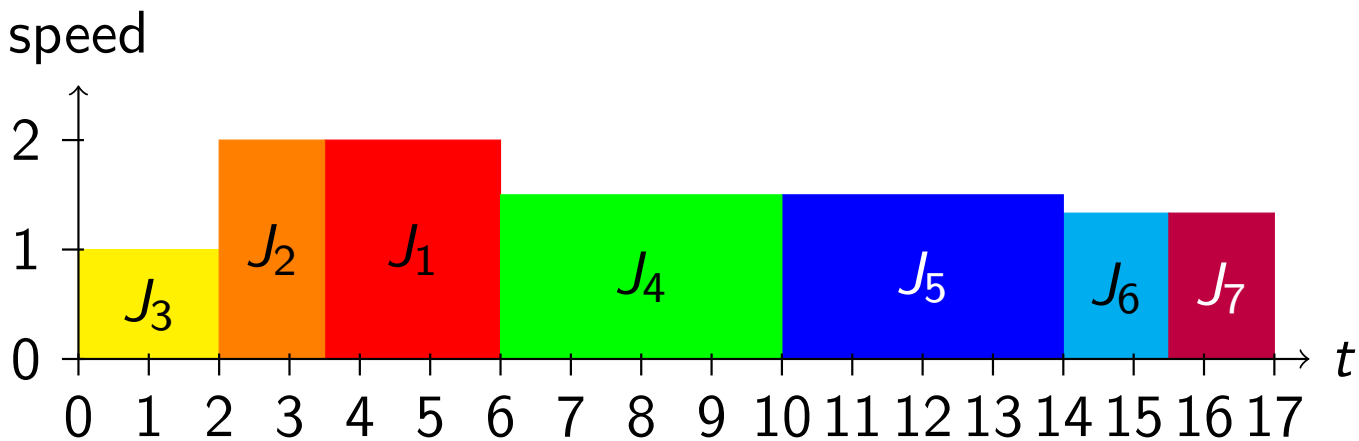
J_3		J_5		J_7	
orig	new	orig	new	orig	new
$(0, 4, 2)$		$(7, 13, 12)$		$(8, 13, 12)$	



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
$(0, 4, 2)$	$(0, 2, 2)$	$(7, 13, 12)$	$(2, 5, 2)$	$(8, 13, 12)$	$(2, 5, 2)$

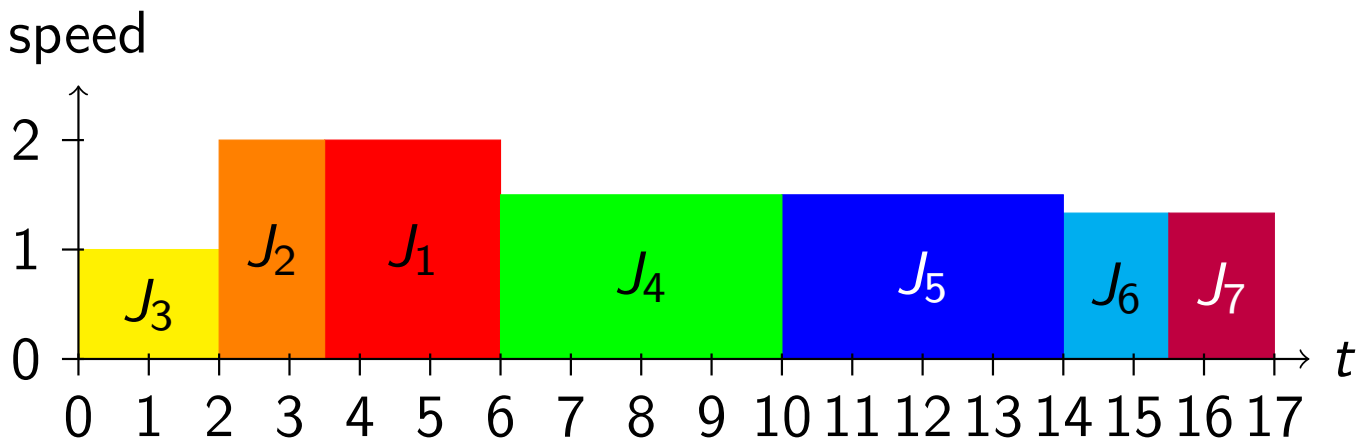


YDS Algorithm

- 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.



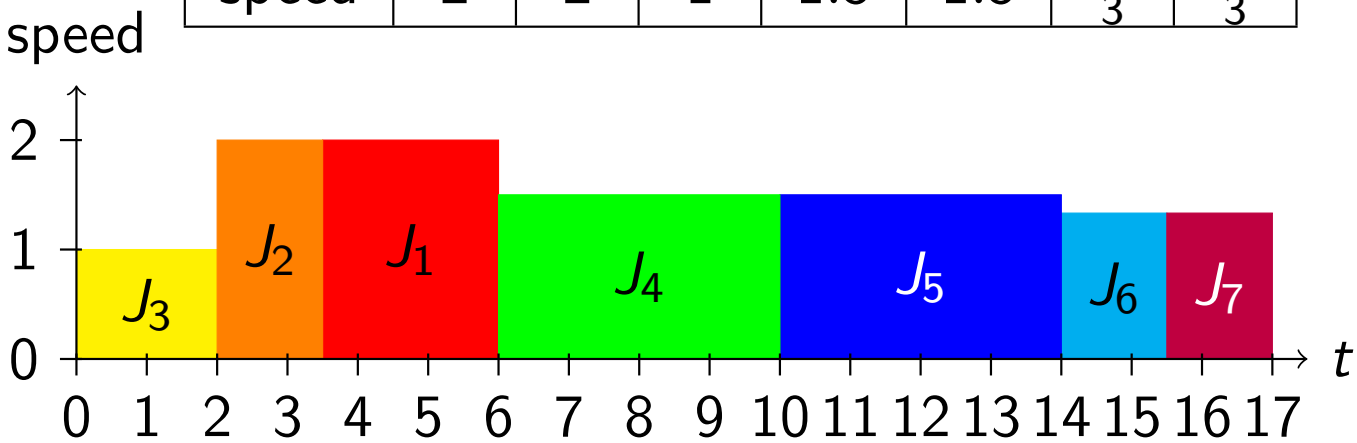
YDS Algorithm

- 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

YDS Algorithm

Discrete Speeds

Energy and Power Optimization for Multiprocessor Systems

Appendix for SFA

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

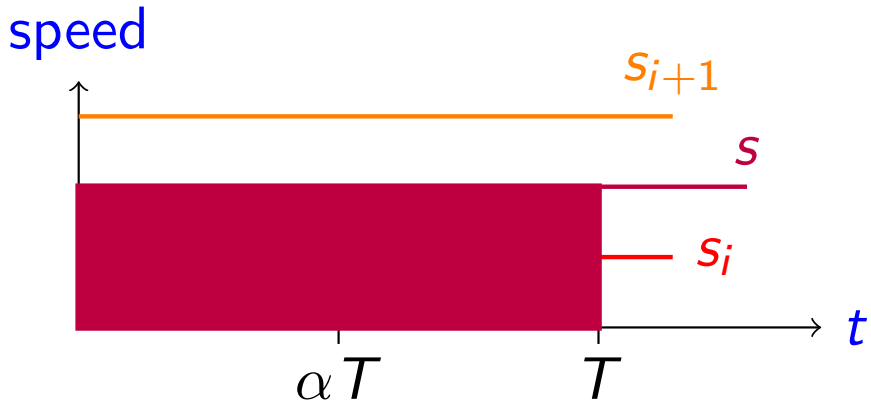
- Intel's SpeedStep - 2 speeds
- AMD's PowerNow - 9 speeds
- Intel's Foxton technology - 64 speeds

Applying YDS Algorithm

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

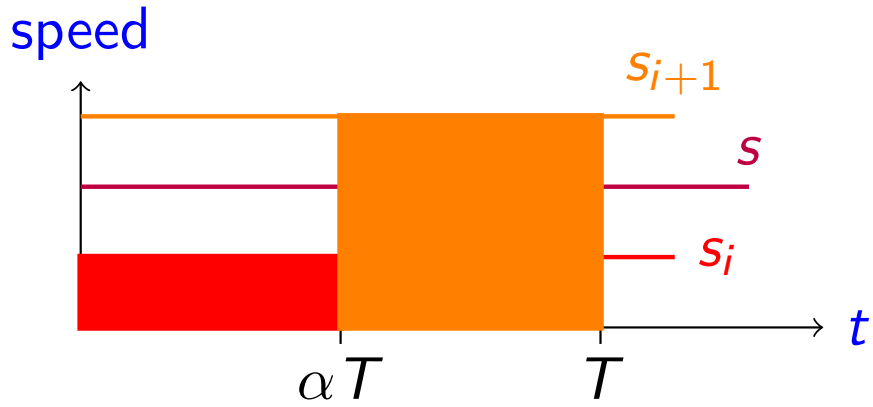
Our objective here: reduce the time complexity to $O(K|\mathcal{J}|\log|\mathcal{J}|)$ for systems with K discrete speeds.

Linear Combination of Speeds



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

Linear Combination of Speeds

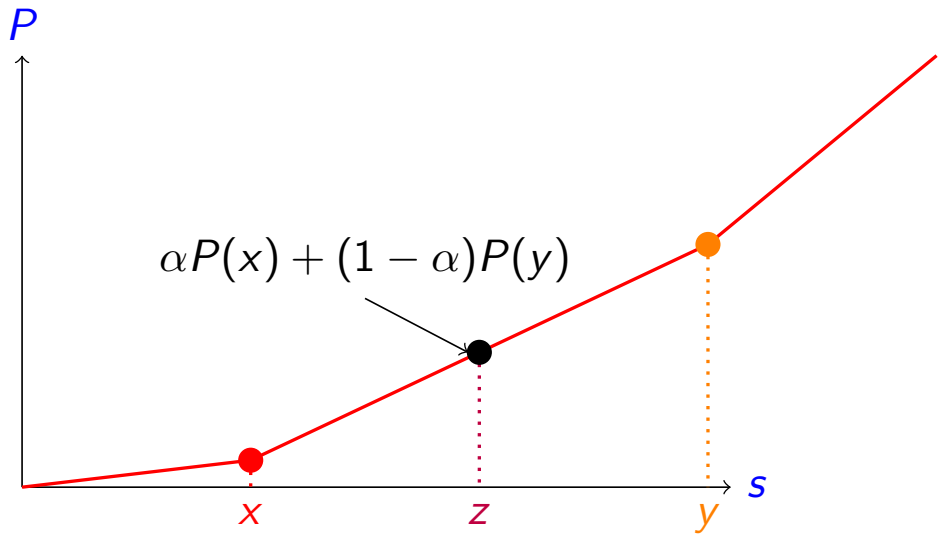


- The optimal solution is to execute at speed s for T time units
- The optimal solution for systems with discrete speeds is to execute at two closest speeds $s_i < s \leq s_{i+1}$ such that

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

  - 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.

Outline

YDS Algorithm

Discrete Speeds

Energy and Power Optimization for Multiprocessor Systems

Appendix for SFA

Low-Power Methodologies

Dynamic Power Management (DPM):

- Technique for putting cores in a low-power mode: idle, sleep, off, etc.

Low-Power Methodologies

Dynamic Power Management (DPM):

- Technique for putting cores in a low-power mode: idle, sleep, off, etc.

Dynamic Voltage and Frequency Scaling (DVFS):

- Technique for scaling the voltage and frequency of cores.

Low-Power Methodologies

Dynamic Power Management (DPM):

- Technique for putting cores in a low-power mode: idle, sleep, off, etc.

Dynamic Voltage and Frequency Scaling (DVFS):

- Technique for scaling the voltage and frequency of cores.
- Per-core DVFS:
  - Individual voltage and frequency for cores.
  - Optimal, but too expensive to manufacture.

Low-Power Methodologies

Dynamic Power Management (DPM):

- Technique for putting cores in a low-power mode: idle, sleep, off, etc.

Dynamic Voltage and Frequency Scaling (DVFS):

- Technique for scaling the voltage and frequency of cores.
- Per-core DVFS:
  - Individual voltage and frequency for cores.
  - Optimal, but too expensive to manufacture.
- Global DVFS:
  - All cores share the same voltage.
  - Energy inefficient.

Low-Power Methodologies

Dynamic Voltage and Frequency Scaling (DVFS):

- Multiple Voltage Islands:
  - Compromise between *Per-core DVFS* and *Global DVFS*.
  - Cores are grouped into *Voltage Islands*.
  - Islands can have different voltages.

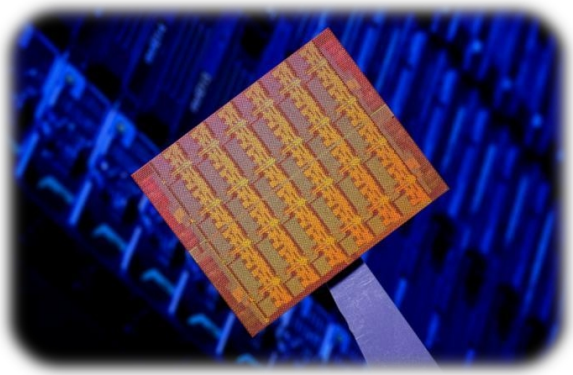


Figure : Intel's SCC snapshot

SFA: Introduction

CMOS-core Power Model

$$P(s) = P_{\text{dynamic}}(s) + P_{\text{static}}$$

SFA: Introduction

CMOS-core Power Model

$$P(s) = P_{\text{dynamic}}(s) + P_{\text{static}}$$

Considering that:

$$P_{\text{dynamic}}(s) = C_{\text{eff}} V_{dd}^2 s$$

$$s \propto \frac{(V_{dd} - V_t)^2}{V_{dd}}$$

SFA: Introduction

CMOS-core Power Model

$$P(s) = P_{\text{dynamic}}(s) + P_{\text{static}}$$

Considering that:

$$P_{\text{dynamic}}(s) = C_{\text{eff}} V_{dd}^2 s$$

$$s \propto \frac{(V_{dd} - V_t)^2}{V_{dd}}$$

We can approximate to:

$$P(s) = \alpha s^{\gamma} + \beta$$

SFA: Introduction

CMOS-core Power Model

$$P(s) = P_{\text{dynamic}}(s) + P_{\text{static}}$$

Considering that:

$$P_{\text{dynamic}}(s) = C_{\text{eff}} V_{dd}^2 s$$

$$s \propto \frac{(V_{dd} - V_t)^2}{V_{dd}}$$

We can approximate to:

$$P(s) = \alpha s^{\gamma} + \beta$$

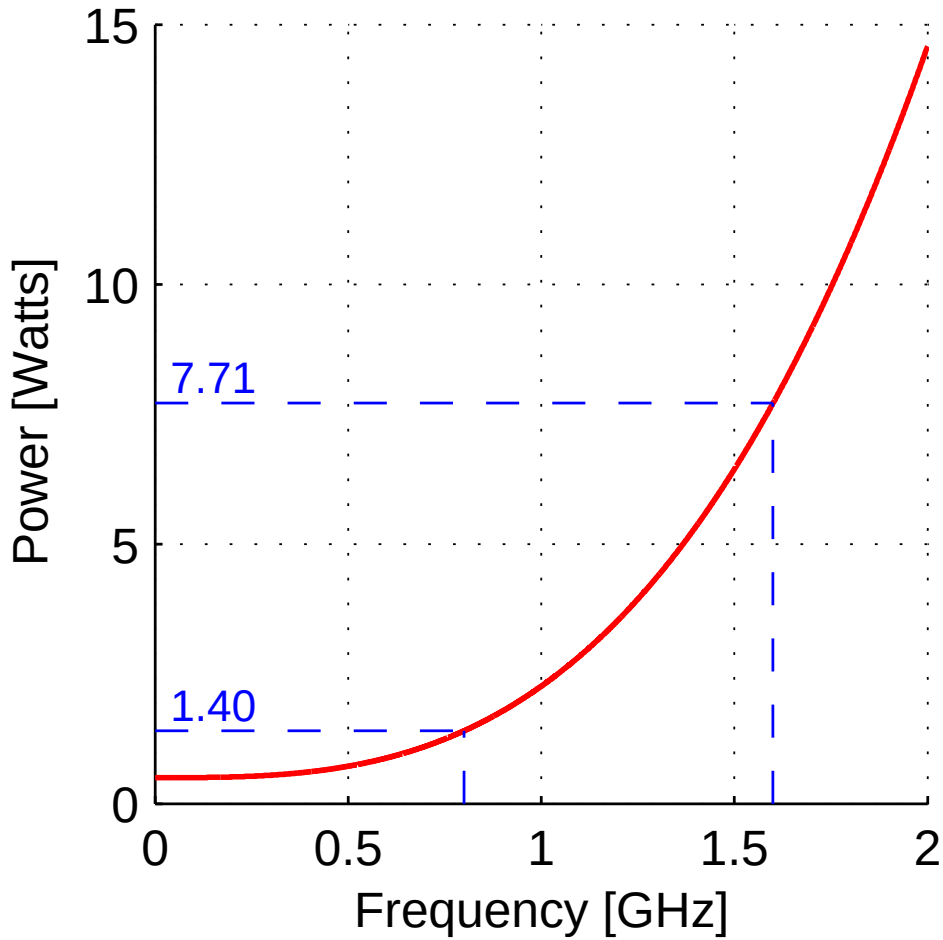


Figure : $\alpha = 1.76 \frac{\text{Watts}}{\text{GHz}^3}$, $\gamma = 3$ and $\beta = 0.5$ Watts

SFA: Introduction

Energy Consumption

$$E(s) = (\alpha s^{\gamma} + \beta)\frac{\Delta c}{s}$$

Critical Frequency:

$$s_{\text{crit}} = \sqrt[\gamma]{\frac{\beta}{(\gamma - 1)\,\alpha}}$$

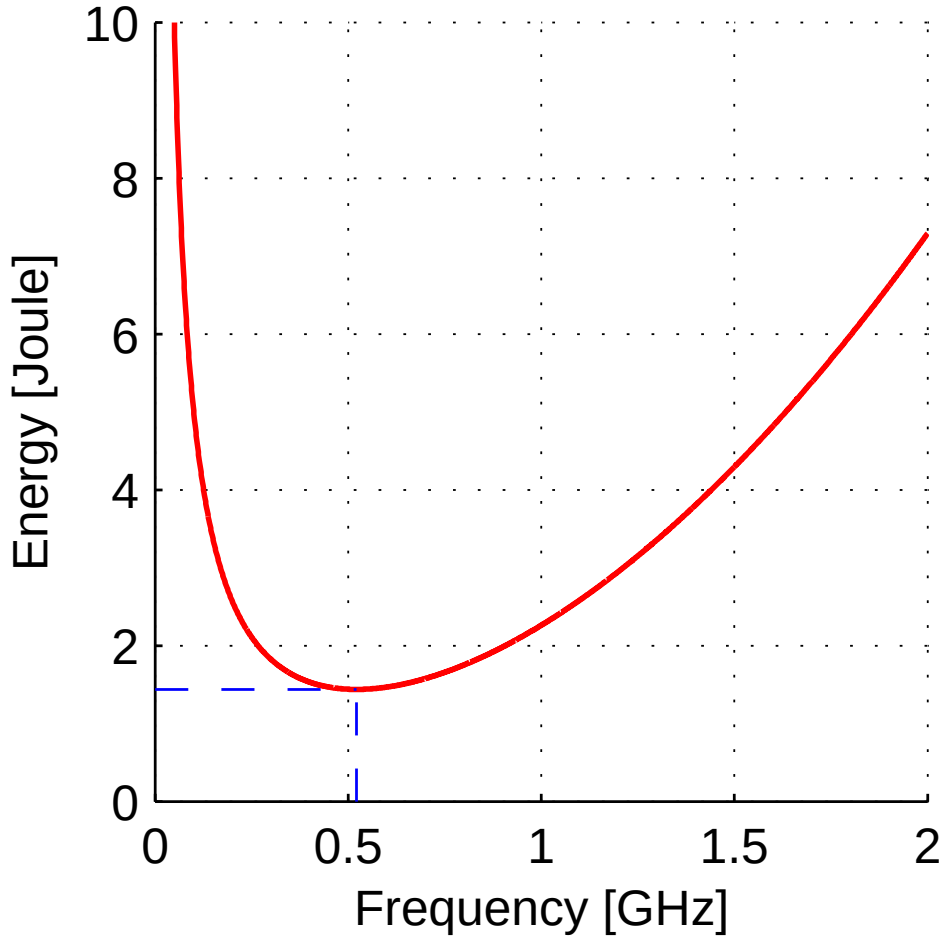


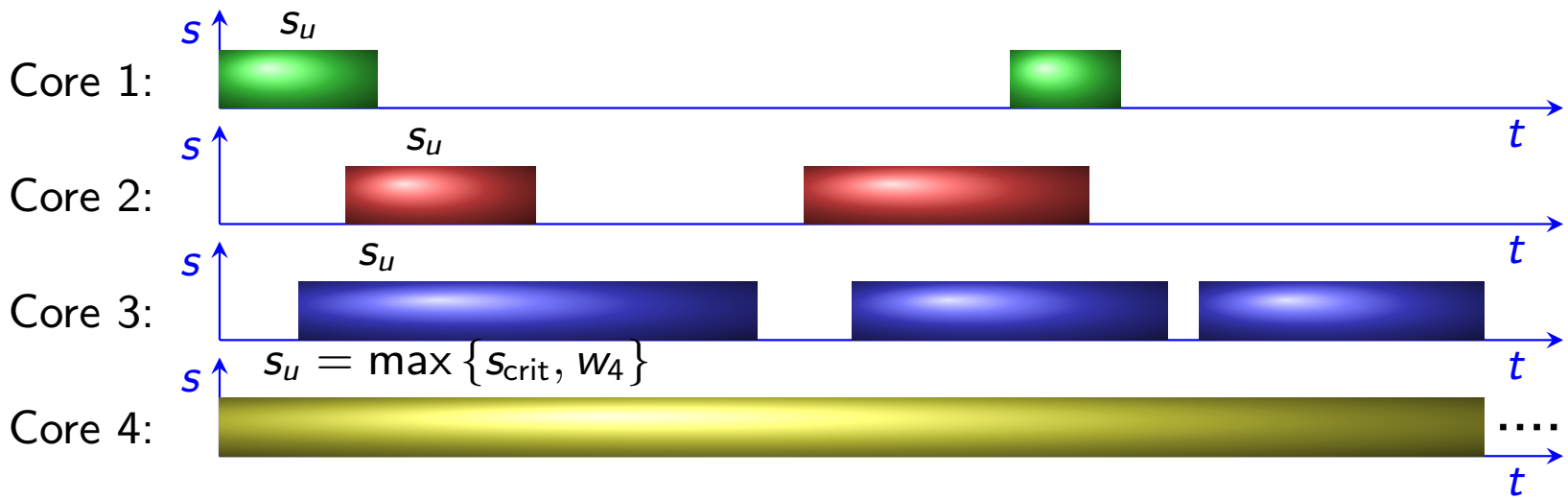
Figure : $\alpha = 1.76\frac{\text{Watts}}{\text{GHz}^3}$, $\gamma = 3$, $\beta = 0.5$ Watts and $\Delta c = 10^9$ cycles

SFA: Motivation

DVFS Schedule $\rightarrow$ Single Frequency Approximation (SFA)

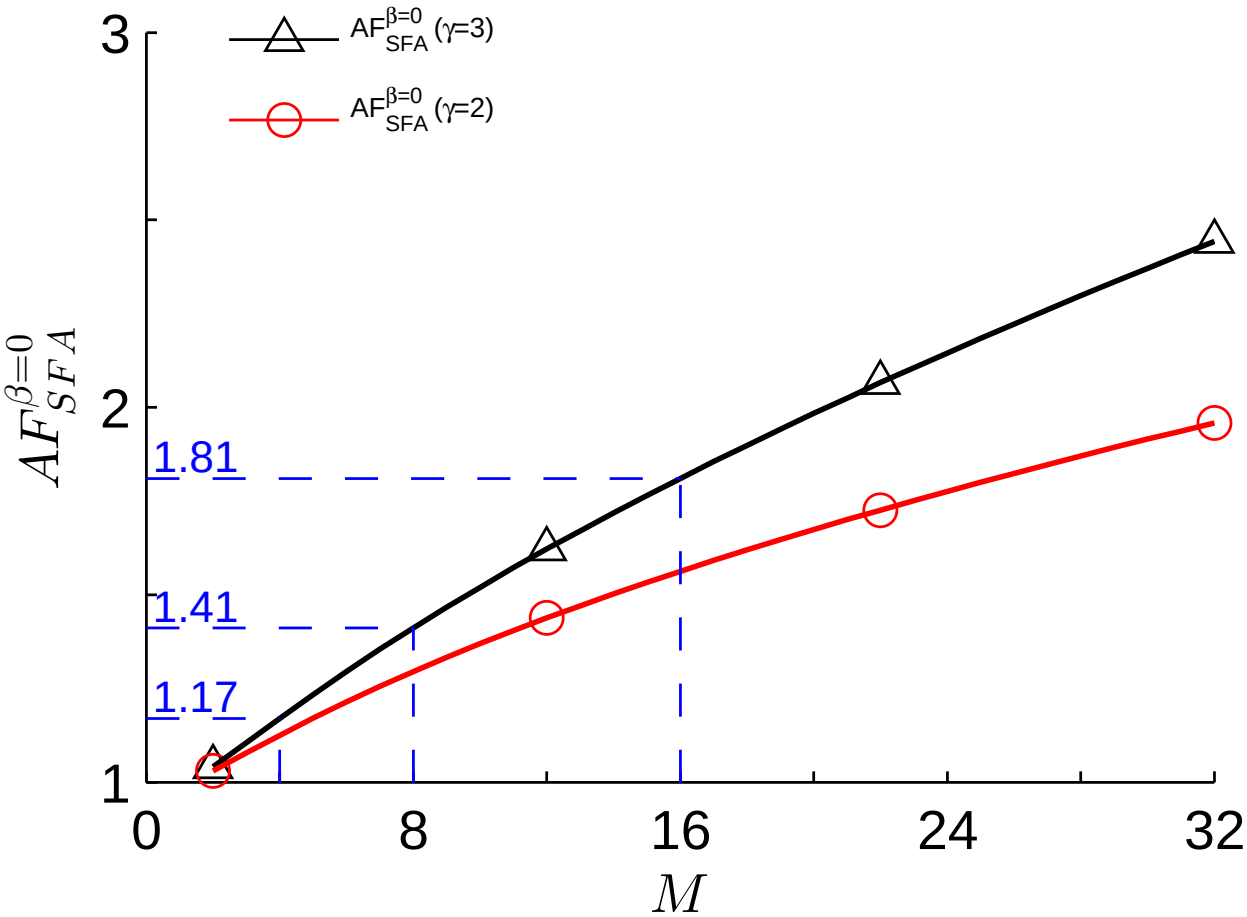
Scheme:

- Use the lowest voltage/frequency, satisfying the timing constraints.
- Linear time complexity. Is the *simplest* and *most intuitive* strategy.
- Not optimal, but significantly reduces the management overhead.
- No frequency alignment between cores $\Longrightarrow$ Any uni-core DPM technique can be adopted individually in each core.



SFA: Negligible Static Power

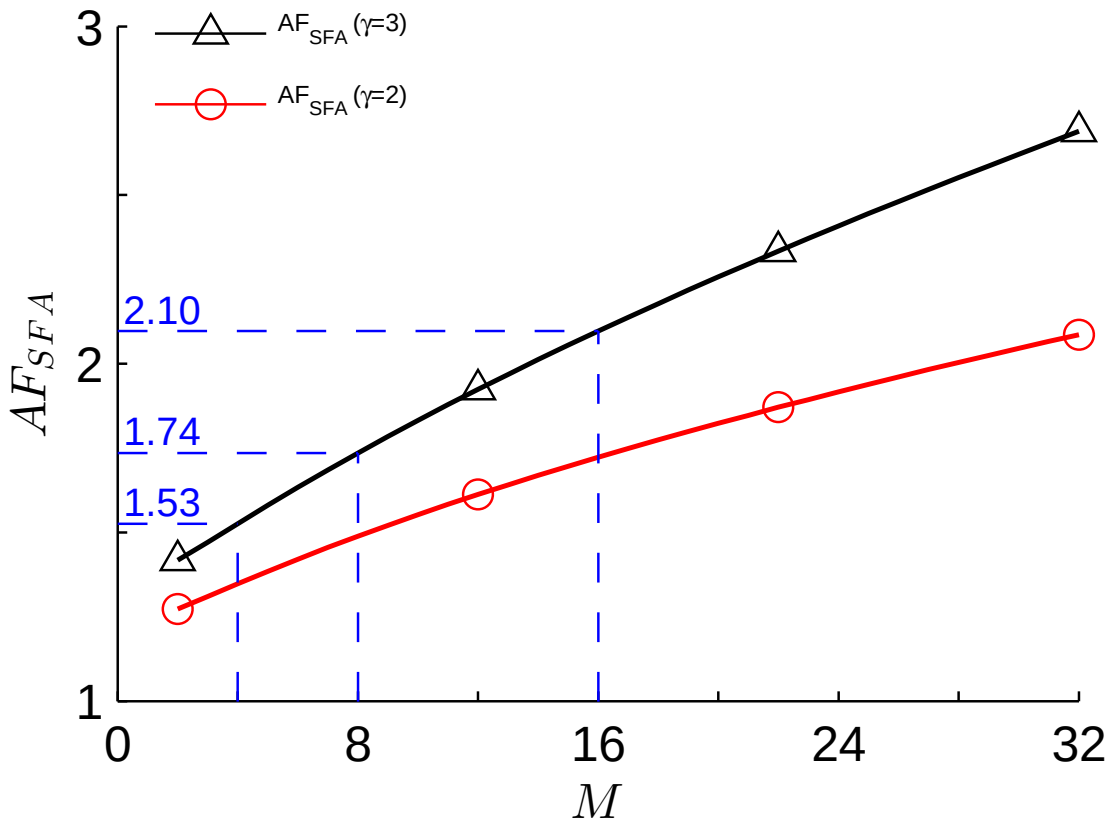
Approximation factor of SFA when $\beta = 0$ (function of M):



Note: $AF_{SFA}^{\beta=0}$ *only depends on the values of* γ *and* M.

SFA: Non-negligible Static Power

Approximation factor of SFA when $\beta \neq 0 \Rightarrow$



Note: AF_{SFA} only depends on the values of γ and M.

DLTF & SFA: Motivation

In each voltage island (or Global DVFS), for energy minimization:

Task partitioning and DVFS schedule:
Play a major role in energy minimization.

DLTF & SFA: Motivation

In each voltage island (or Global DVFS), for energy minimization:

Task partitioning and DVFS schedule:
Play a major role in energy minimization.

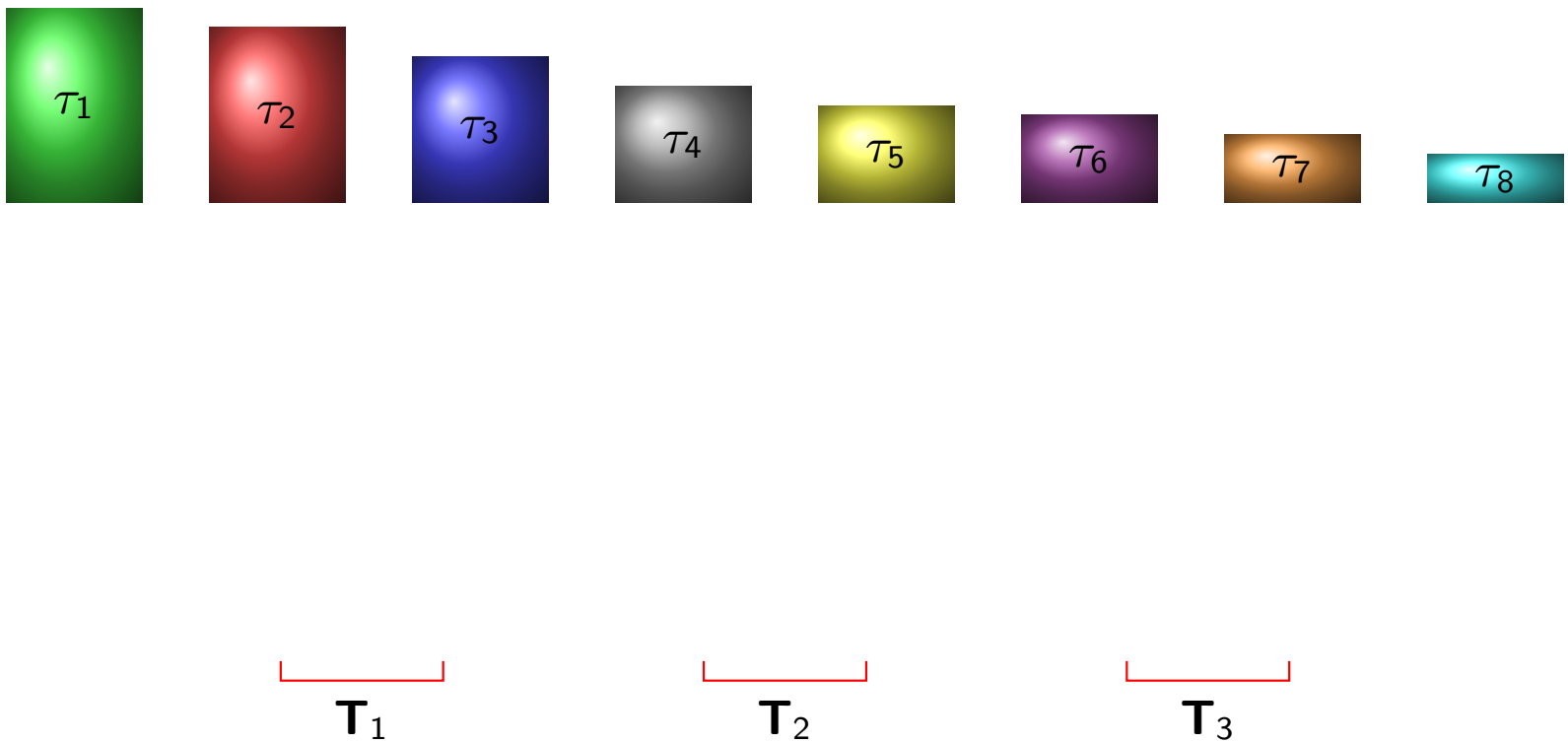
Task Partitioning:

- Convexity of $E(s)$: For the same workload $\Rightarrow$ $2 \cdot E(s) \leq E(2 \cdot s)$.
- In general, load balancing reduces the dynamic energy consumption.
- Balanced solution: with high complexity and not always feasible.
- Good option: polynomial time algorithm based on load balancing, e.g., *Largest-Task-First* (LTF) strategy.

DLTF & SFA: Motivation

Largest-Task-First (LTF) Strategy Example:

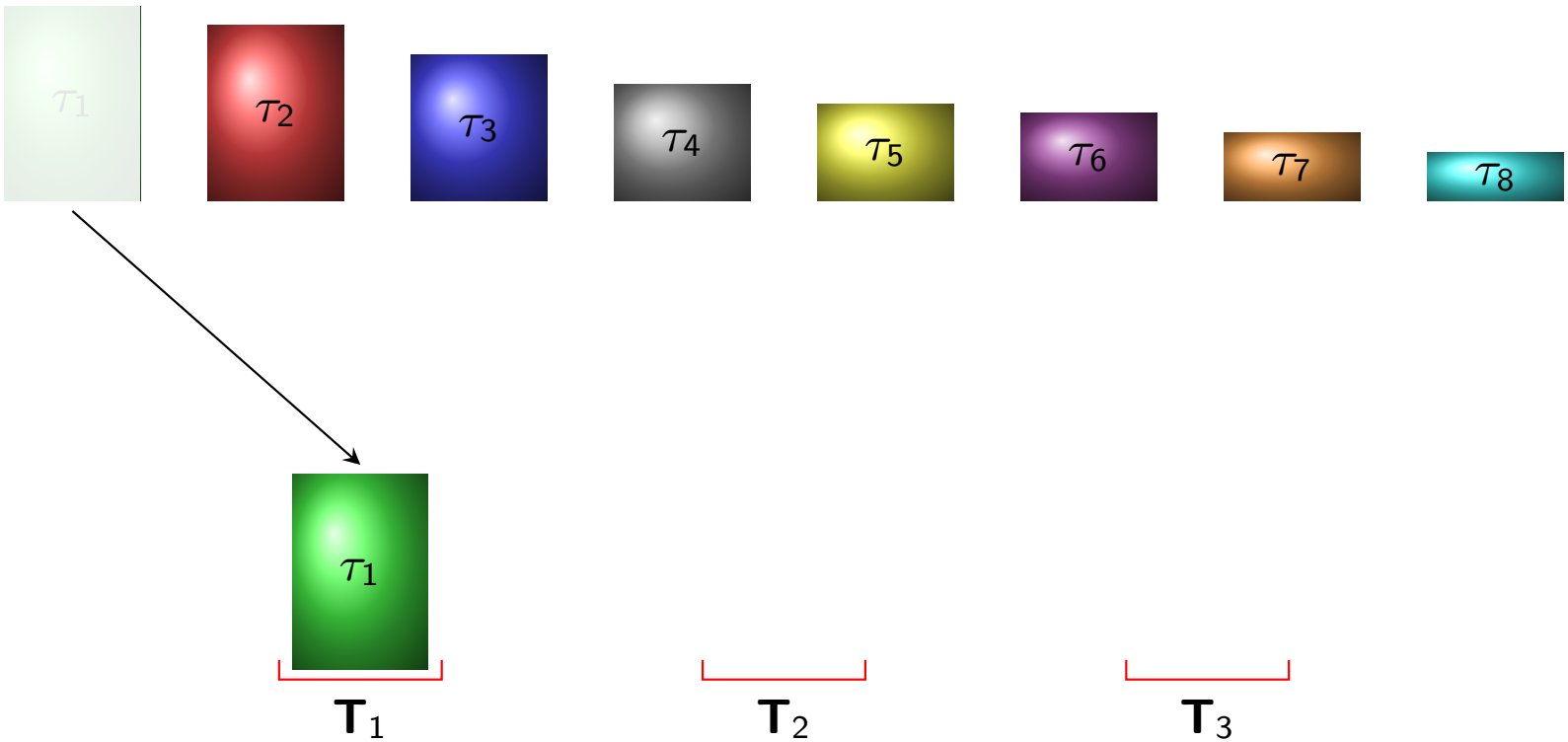
- 8 tasks: $\{\tau_1, \tau_2, \ldots, \tau_8\}$
- Partitioned into 3 task sets: $\{\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3\}$



DLTF & SFA: Motivation

Largest-Task-First (LTF) Strategy Example:

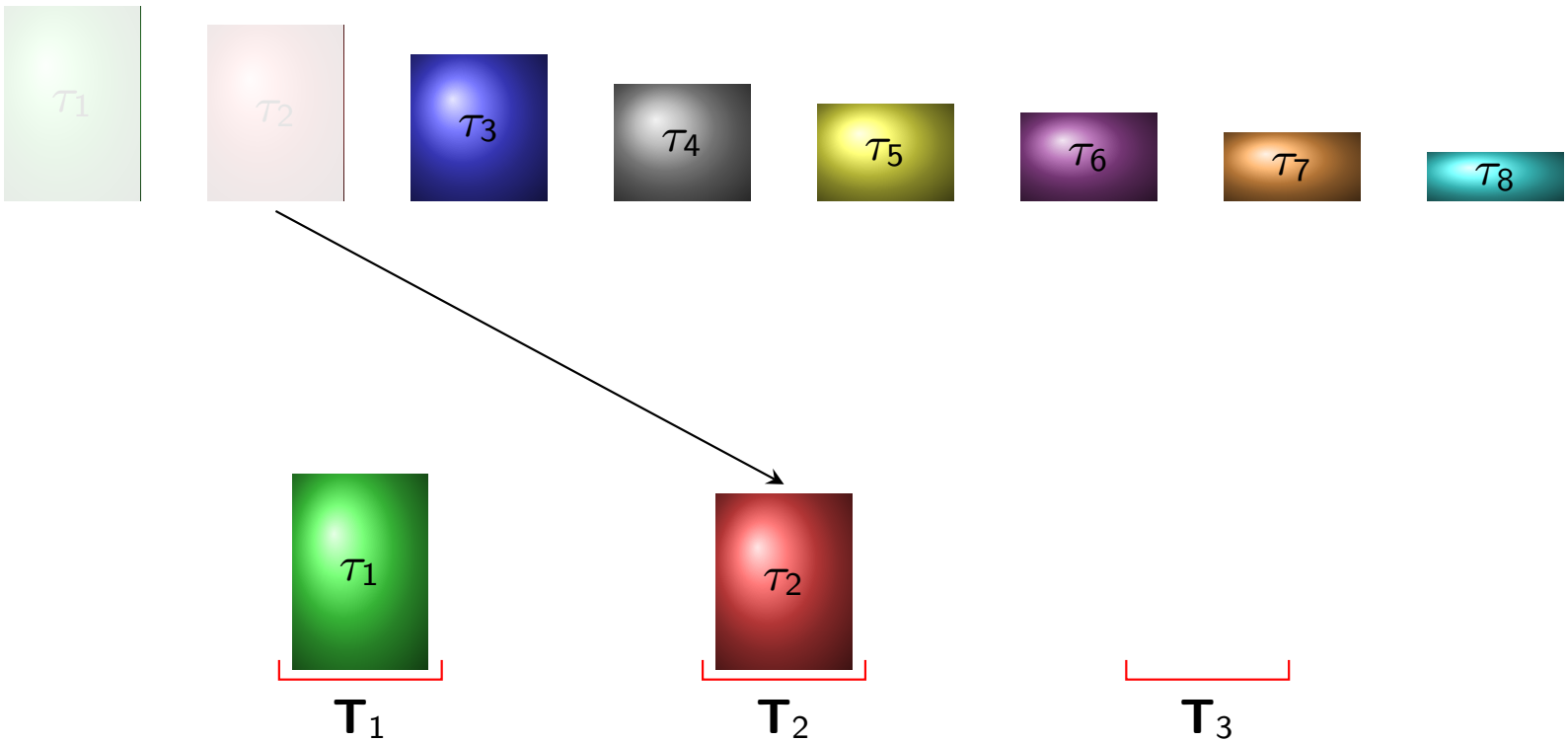
- 8 tasks: $\{\tau_1, \tau_2, \ldots, \tau_8\}$
- Partitioned into 3 task sets: $\{\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3\}$



DLTF & SFA: Motivation

Largest-Task-First (LTF) Strategy Example:

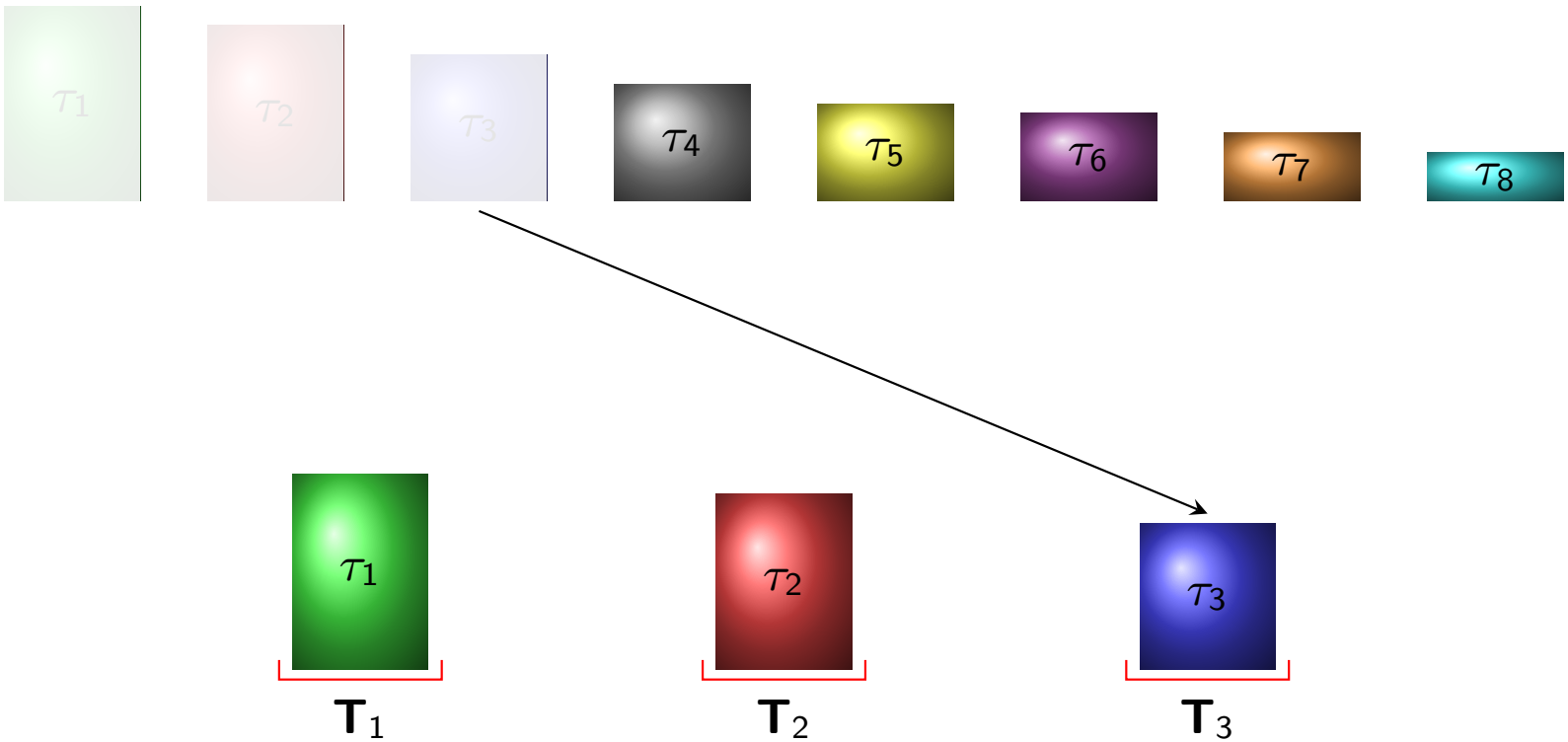
- 8 tasks: $\{\tau_1, \tau_2, \ldots, \tau_8\}$
- Partitioned into 3 task sets: $\{\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3\}$



DLTF & SFA: Motivation

Largest-Task-First (LTF) Strategy Example:

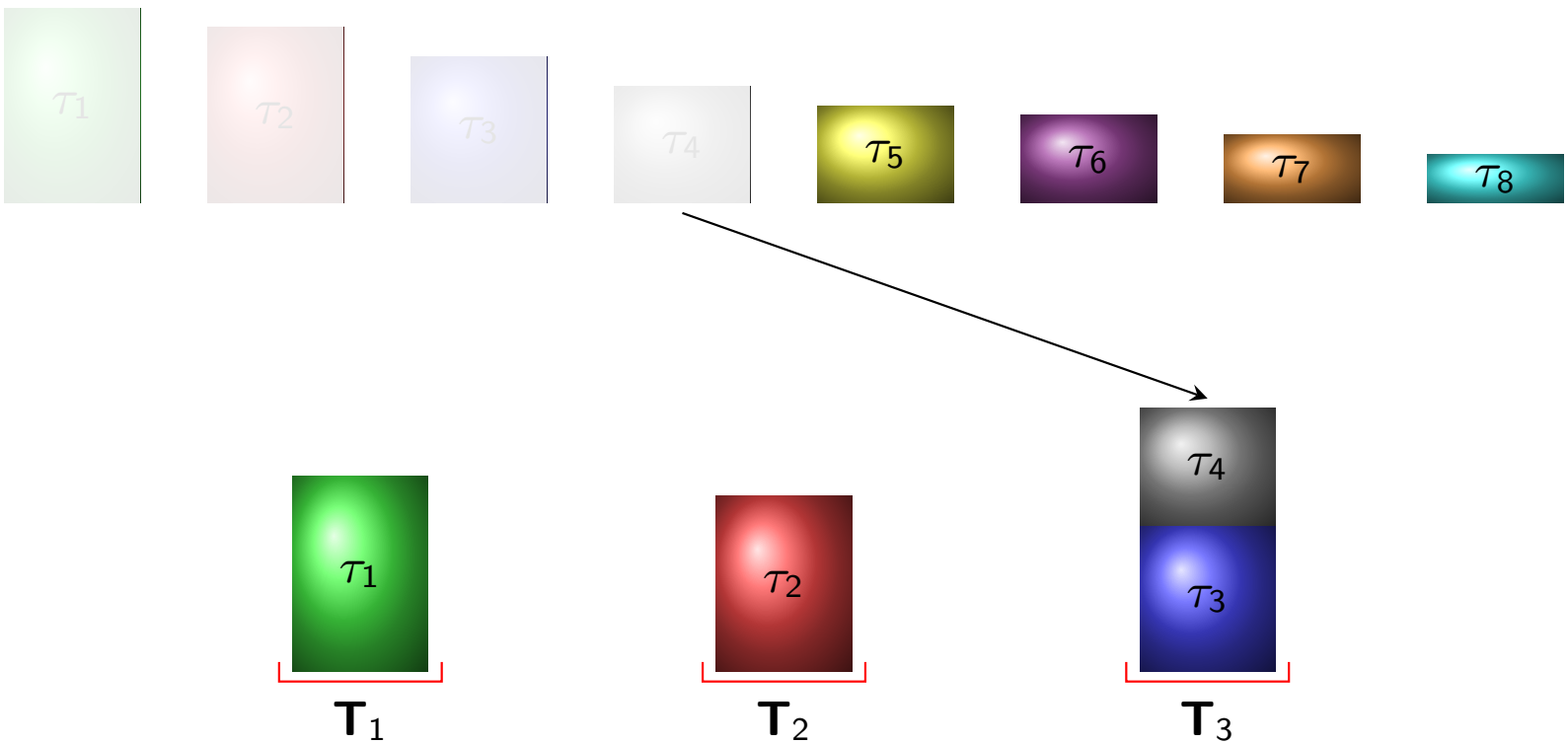
- 8 tasks: $\{\tau_1, \tau_2, \ldots, \tau_8\}$
- Partitioned into 3 task sets: $\{\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3\}$



DLTF & SFA: Motivation

Largest-Task-First (LTF) Strategy Example:

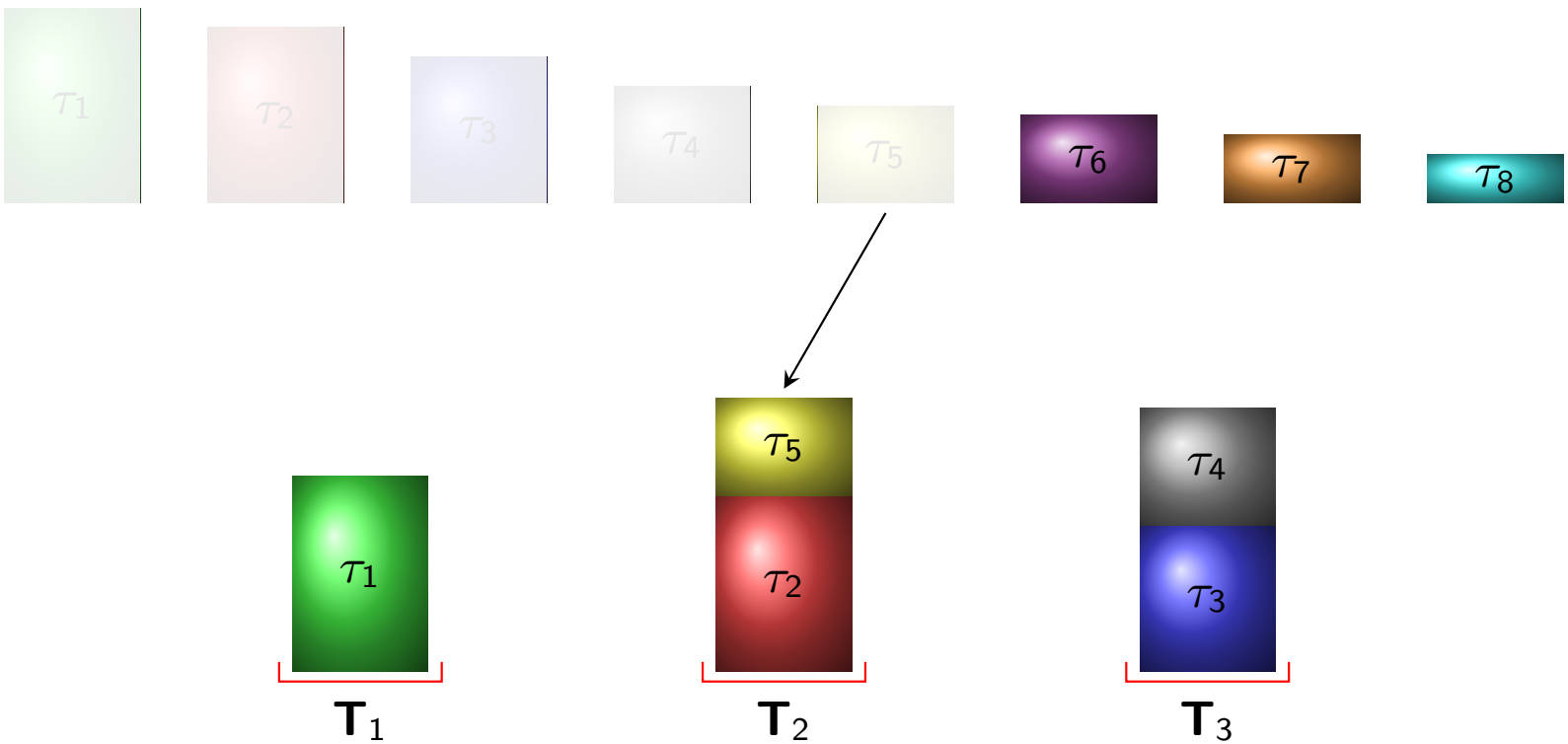
- 8 tasks: $\{\tau_1, \tau_2, \ldots, \tau_8\}$
- Partitioned into 3 task sets: $\{\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3\}$



DLTF & SFA: Motivation

Largest-Task-First (LTF) Strategy Example:

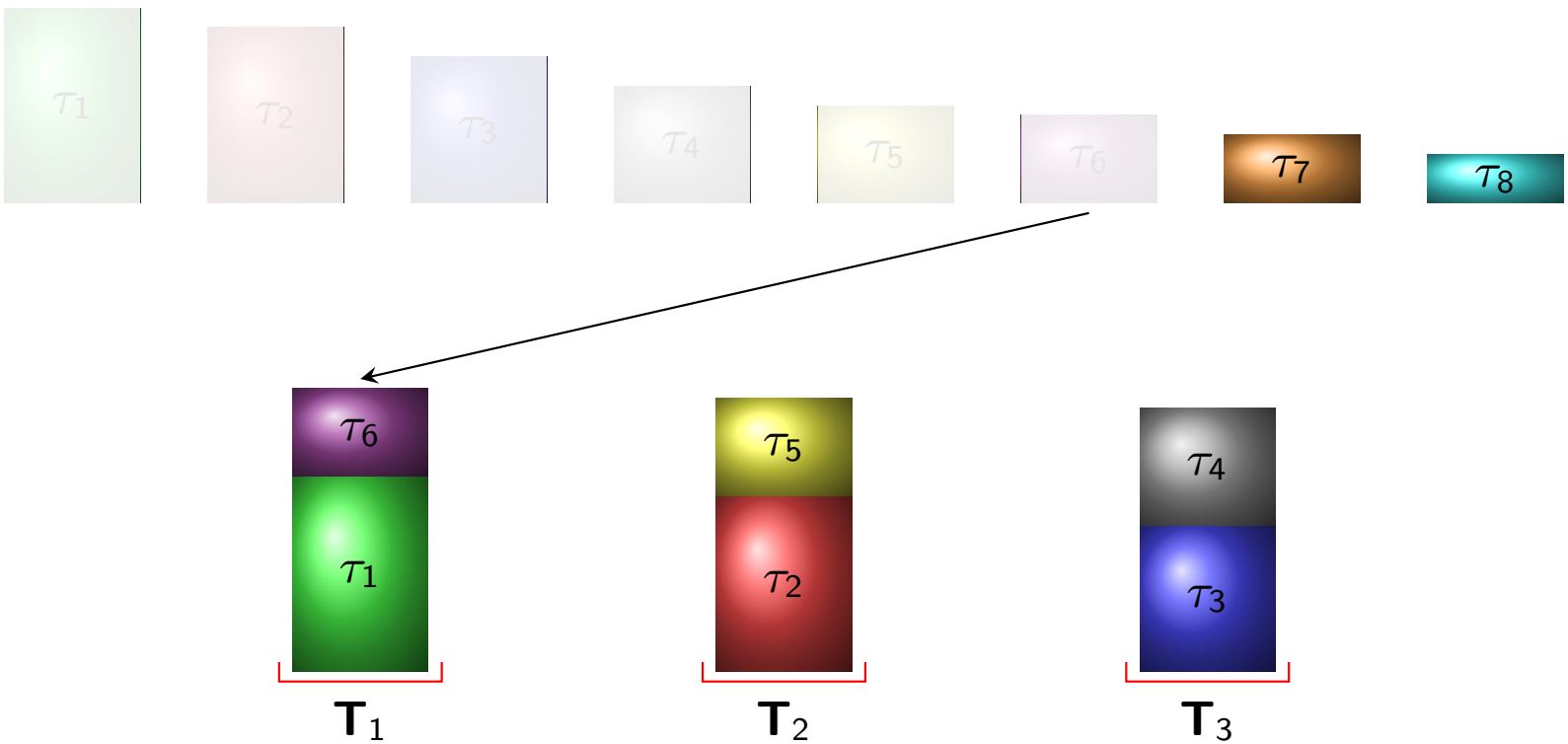
- 8 tasks: $\{\tau_1, \tau_2, \ldots, \tau_8\}$
- Partitioned into 3 task sets: $\{\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3\}$



DLTF & SFA: Motivation

Largest-Task-First (LTF) Strategy Example:

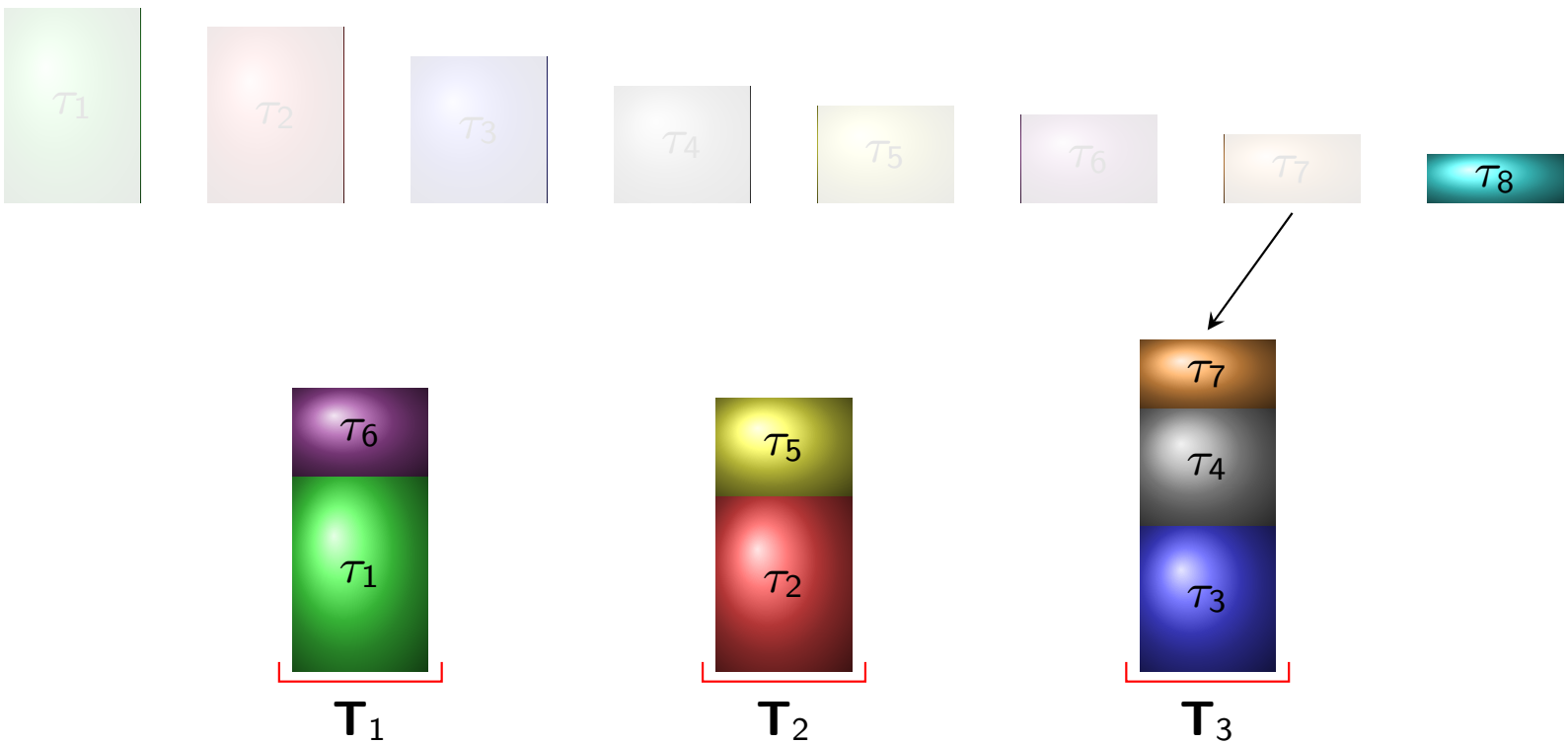
- 8 tasks: $\{\tau_1, \tau_2, \ldots, \tau_8\}$
- Partitioned into 3 task sets: $\{\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3\}$



DLTF & SFA: Motivation

Largest-Task-First (LTF) Strategy Example:

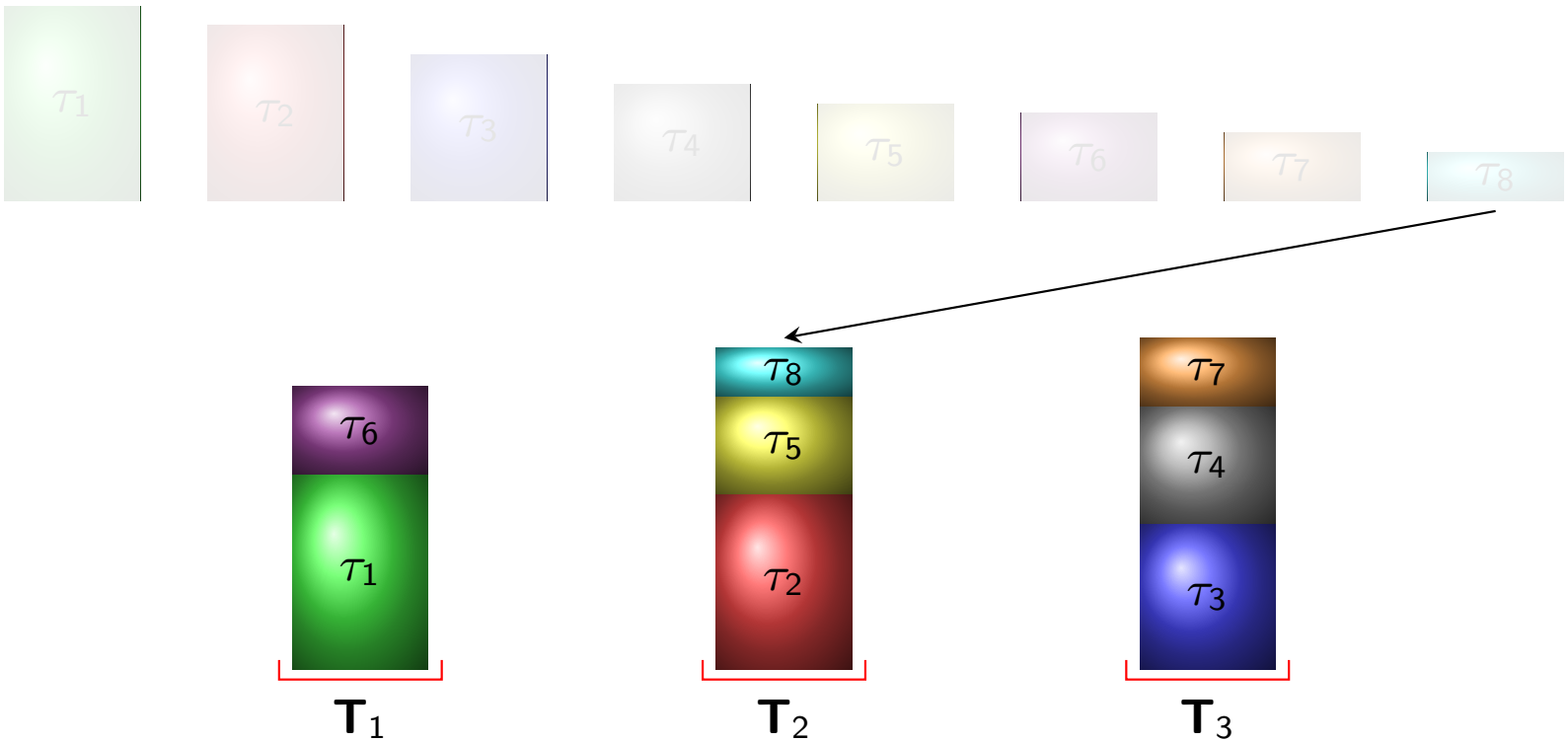
- 8 tasks: $\{\tau_1, \tau_2, \ldots, \tau_8\}$
- Partitioned into 3 task sets: $\{\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3\}$



DLTF & SFA: Motivation

Largest-Task-First (LTF) Strategy Example:

- 8 tasks: $\{\tau_1, \tau_2, \ldots, \tau_8\}$
- Partitioned into 3 task sets: $\{\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3\}$



DLTF & SFA: Motivation

Combining LTF and SFA for energy minimization:

- Is a practical solution.
- Very easy to implement.

What is the worst-case performance in terms of energy efficiency?

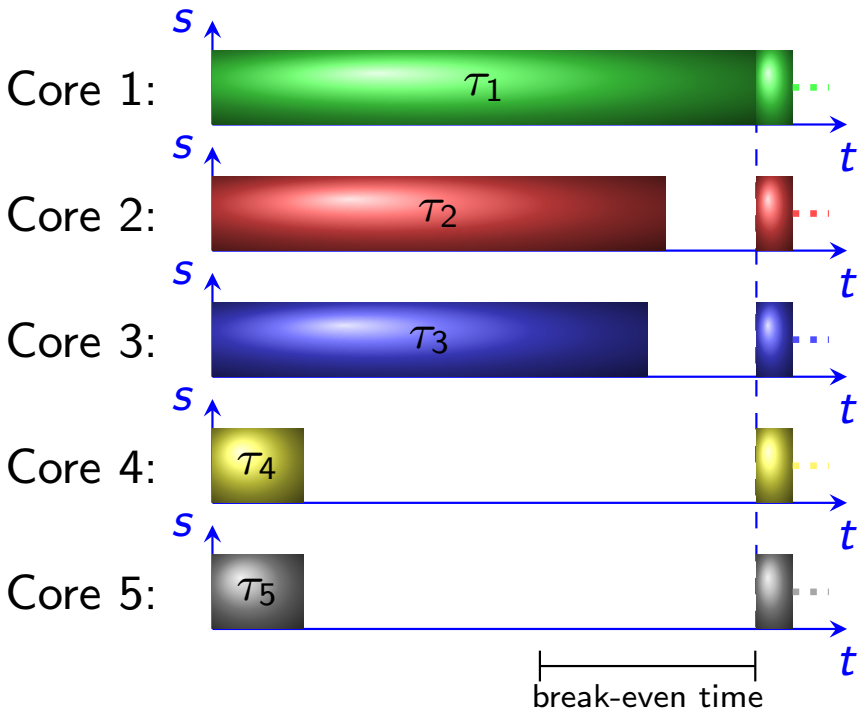
DLTF & SFA: Double-Largest-Task-First

- Initial solution: Task partitioning with LTF.
- Tasks are regrouped, shutting down unnecessary cores.
- This reduces the energy consumption for idling under SFA:

DLTF & SFA: Double-Largest-Task-First

- Initial solution: Task partitioning with LTF.
- Tasks are regrouped, shutting down unnecessary cores.
- This reduces the energy consumption for idling under SFA:

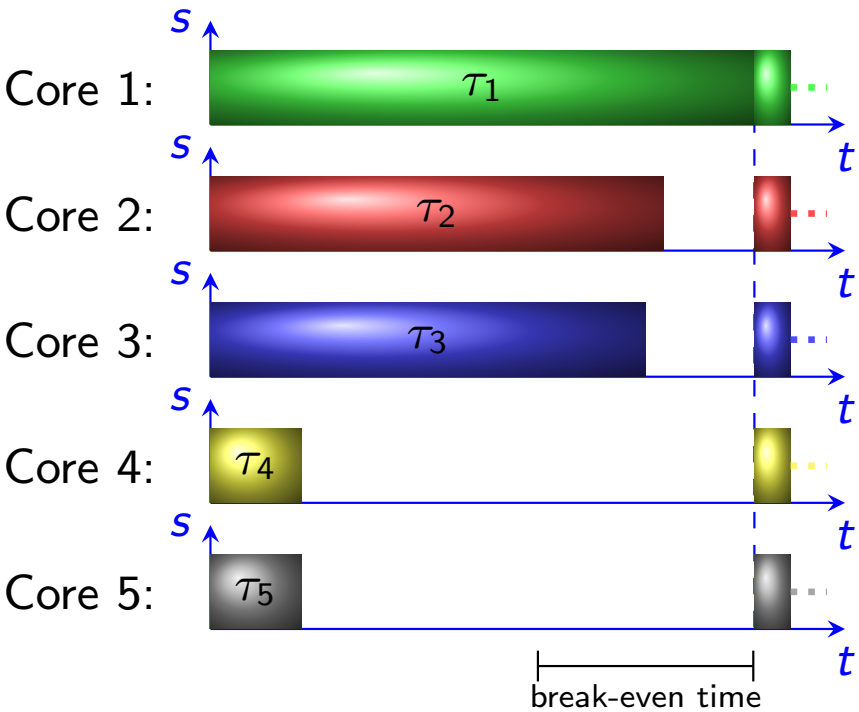
Example: LTF and SFA



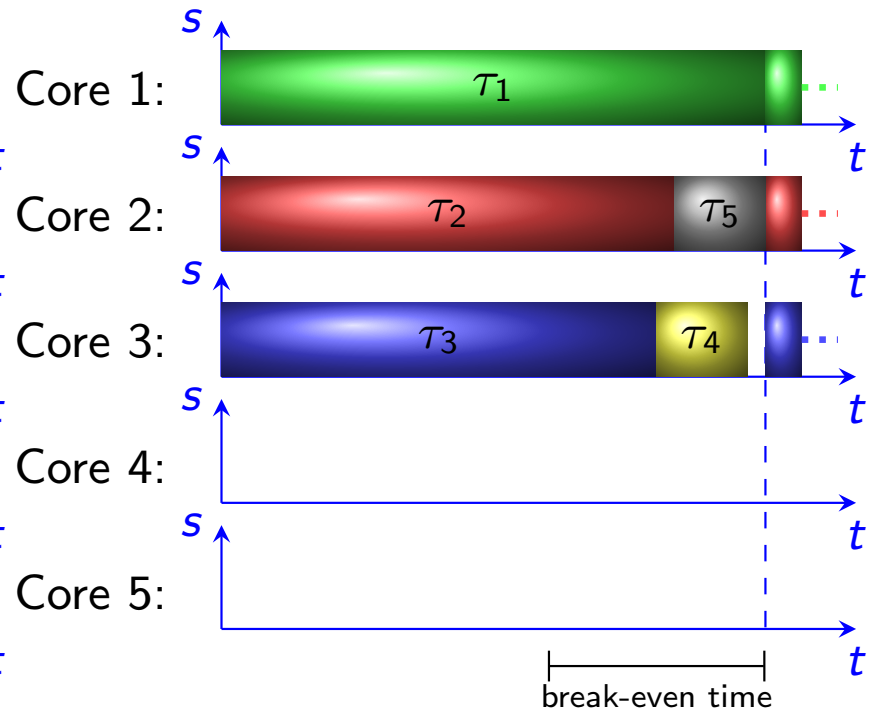
DLTF & SFA: Double-Largest-Task-First

- Initial solution: Task partitioning with LTF.
- Tasks are regrouped, shutting down unnecessary cores.
- This reduces the energy consumption for idling under SFA:

Example: LTF and SFA



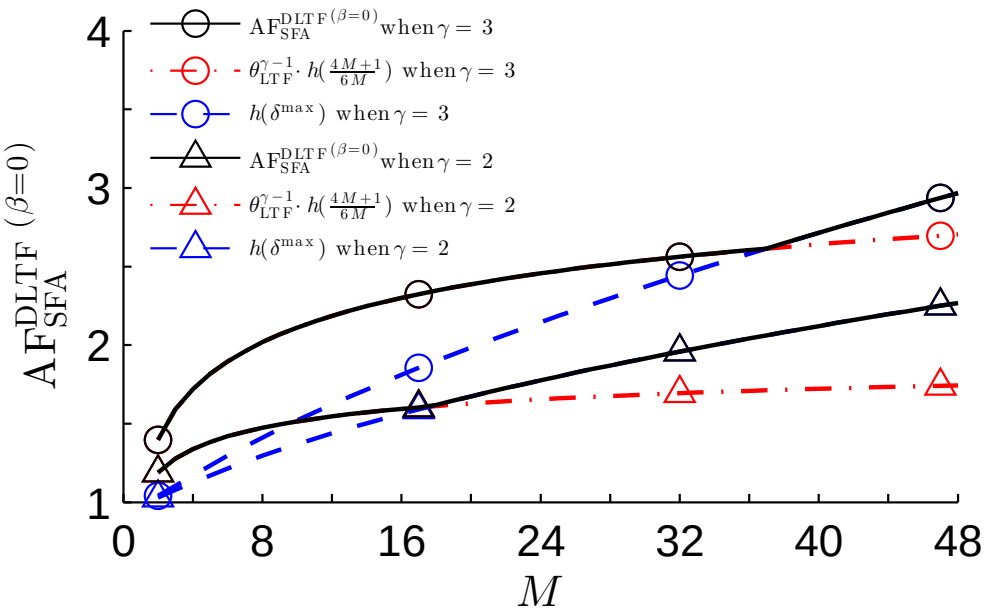
Example: DLTF and SFA



DLTF & SFA: Negligible Static Power

Worst-case Approximation Factor for DLTF and SFA (when $\beta = 0$):

$$\mathrm{AF}_{\mathrm{SFA}}^{\mathrm{DLTF}(\beta=0)} \leq \max\left\{ h\left(\delta^{\max}\right), \theta_{\mathrm{LTF}}^{\gamma-1} \cdot h\left(\frac{4M+1}{6M}\right) \right\}$$



Note: $AF_{SFA}^{DLTF(\beta=0)}$ only depends on the values of γ and M.

Task Mapping

- Consider a system with $V = 8$ voltage islands and $Q = 8$ cores per island ($M = 64$).

Task Mapping

- Consider a system with $V = 8$ voltage islands and $Q = 8$ cores per island ($M = 64$).
- The power consumption function is $P(s) = 2\frac{\text{Watts}}{\text{GHz}^3} \cdot s^3$.

Task Mapping

- Consider a system with $V = 8$ voltage islands and $Q = 8$ cores per island ($M = 64$).
- The power consumption function is $P(s) = 2\frac{\text{Watts}}{\text{GHz}^3} \cdot s^3$.
- Task sets cycle utilizations:
  - $w_{64} = 1GHz$
  - $w_{57} = w_{58} = \ldots = w_{62} = w_{63} = 354MHz$
  - $w_1 = w_2 = \ldots = w_{55} = w_{56} = 0$

Task Mapping

- Consider a system with $V = 8$ voltage islands and $Q = 8$ cores per island ($M = 64$).
- The power consumption function is $P(s) = 2\frac{\text{Watts}}{\text{GHz}^3} \cdot s^3$.
- Task sets cycle utilizations:
  - $w_{64} = 1GHz$
  - $w_{57} = w_{58} = \ldots = w_{62} = w_{63} = 354MHz$
  - $w_1 = w_2 = \ldots = w_{55} = w_{56} = 0$

Assignment 1:
$\{\mathbf{T}_{57}, \mathbf{T}_{58}, \ldots, \mathbf{T}_{64}\} \rightarrow I_8 \Longrightarrow E(s,t) = 6.96$ Joule

Task Mapping

- Consider a system with $V = 8$ voltage islands and $Q = 8$ cores per island ($M = 64$).
- The power consumption function is $P(s) = 2\frac{\text{Watts}}{\text{GHz}^3} \cdot s^3$.
- Task sets cycle utilizations:
  - $w_{64} = 1GHz$
  - $w_{57} = w_{58} = \ldots = w_{62} = w_{63} = 354MHz$
  - $w_1 = w_2 = \ldots = w_{55} = w_{56} = 0$

Assignment 1:
$\{\mathbf{T}_{57}, \mathbf{T}_{58}, \ldots, \mathbf{T}_{64}\} \rightarrow I_8 \Longrightarrow E(s,t) = 6.96$ Joule

Assignment 2:
$\{\mathbf{T}_{57}\} \rightarrow I_1, \{\mathbf{T}_{58}\} \rightarrow I_2, \ldots, \{\mathbf{T}_{64}\} \rightarrow I_8 \Longrightarrow E(s,t) = 2.62$ Joule

Assignment Properties

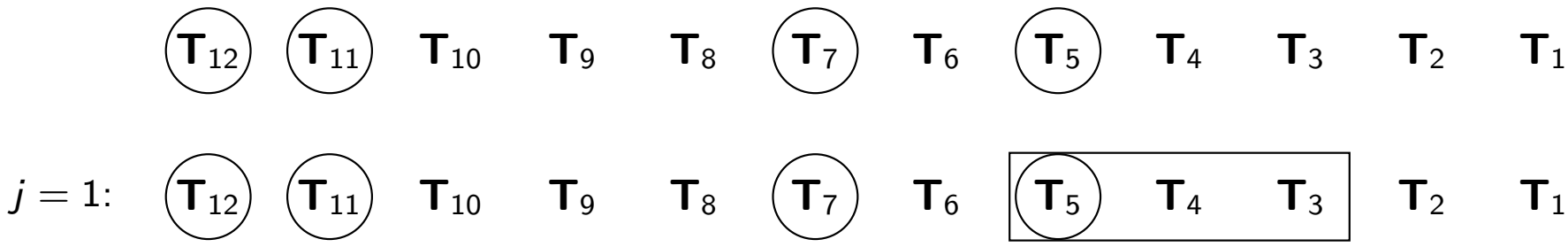
When the *highest utilization task set* on each island is set, for example:

$V = 3$ $Q = 3$ $\mathbf{Y} = \{5, 7, 11, 12\}$

Assignment Properties

When the *highest utilization task set* on each island is set, for example:

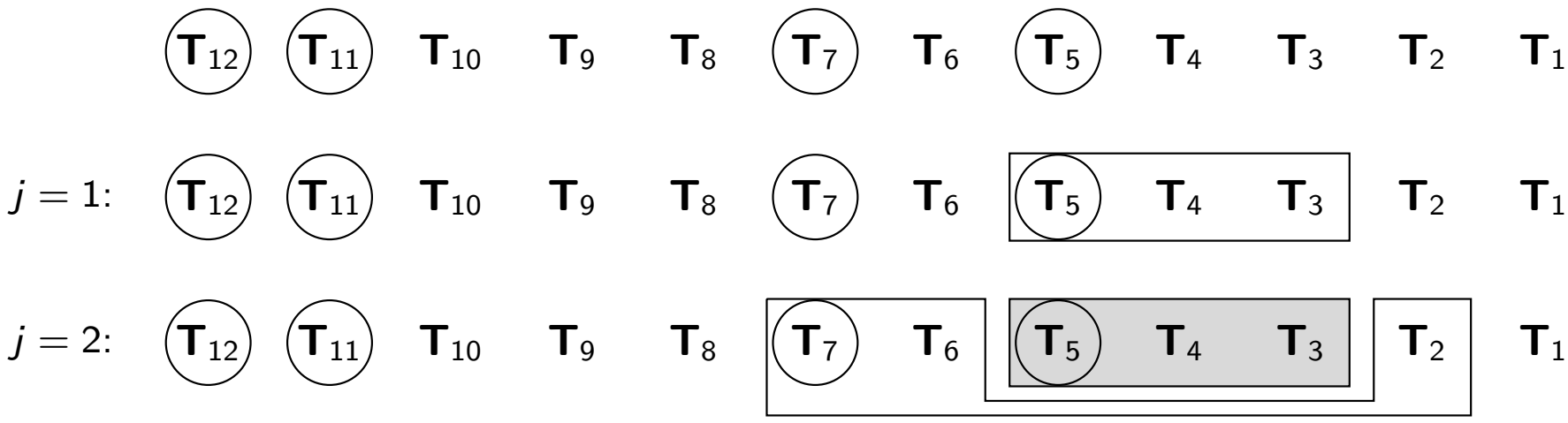
$V = 3$ $Q = 3$ $\mathbf{Y} = \{5, 7, 11, 12\}$



Assignment Properties

When the *highest utilization task set* on each island is set, for example:

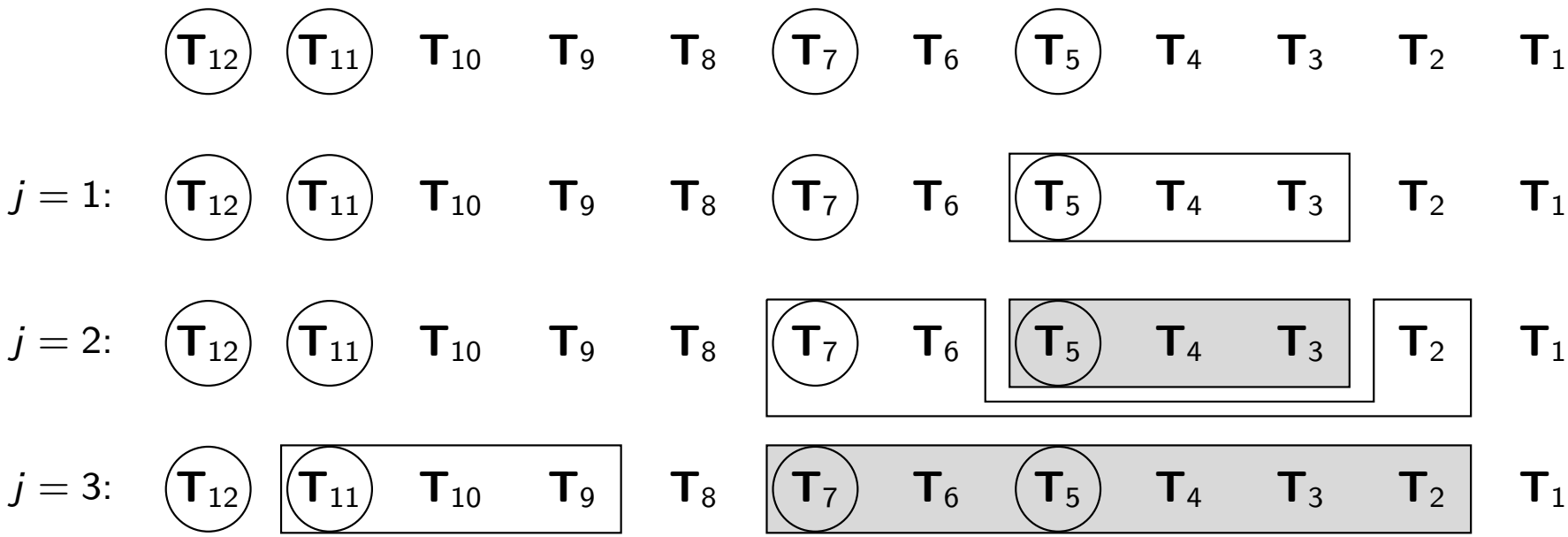
$V = 3$ $Q = 3$ $\mathbf{Y} = \{5, 7, 11, 12\}$



Assignment Properties

When the *highest utilization task set* on each island is set, for example:

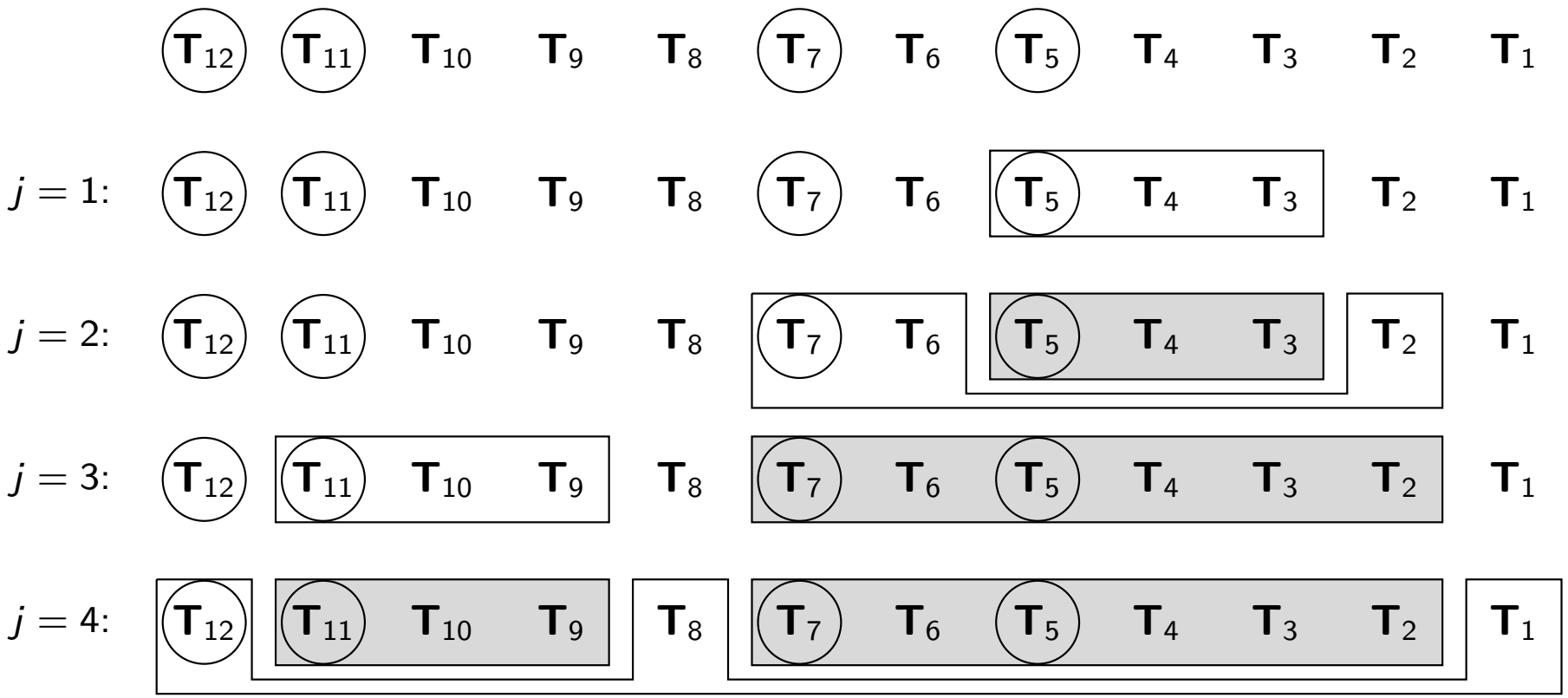
$V = 3$ $Q = 3$ $\mathbf{Y} = \{5, 7, 11, 12\}$



Assignment Properties

When the *highest utilization task set* on each island is set, for example:

$V = 3$ $Q = 3$ $\mathbf{Y} = \{5, 7, 11, 12\}$



Outline

YDS Algorithm

Discrete Speeds

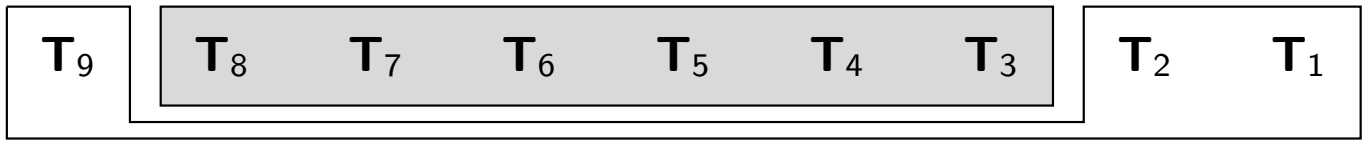
Energy and Power Optimization for Multiprocessor Systems

Appendix for SFA

Dynamic Programming Solution

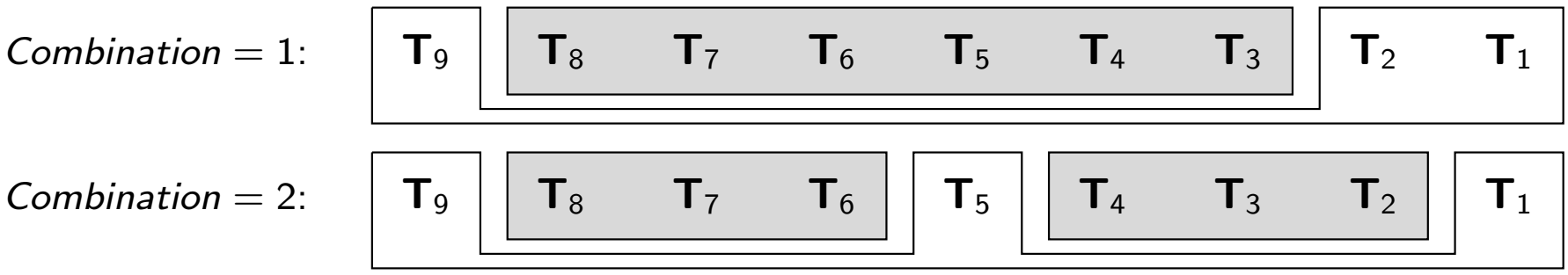
For example, with $V = 3$ and $Q = 3$, to build $G(1, 9)$:

Combination = 1:



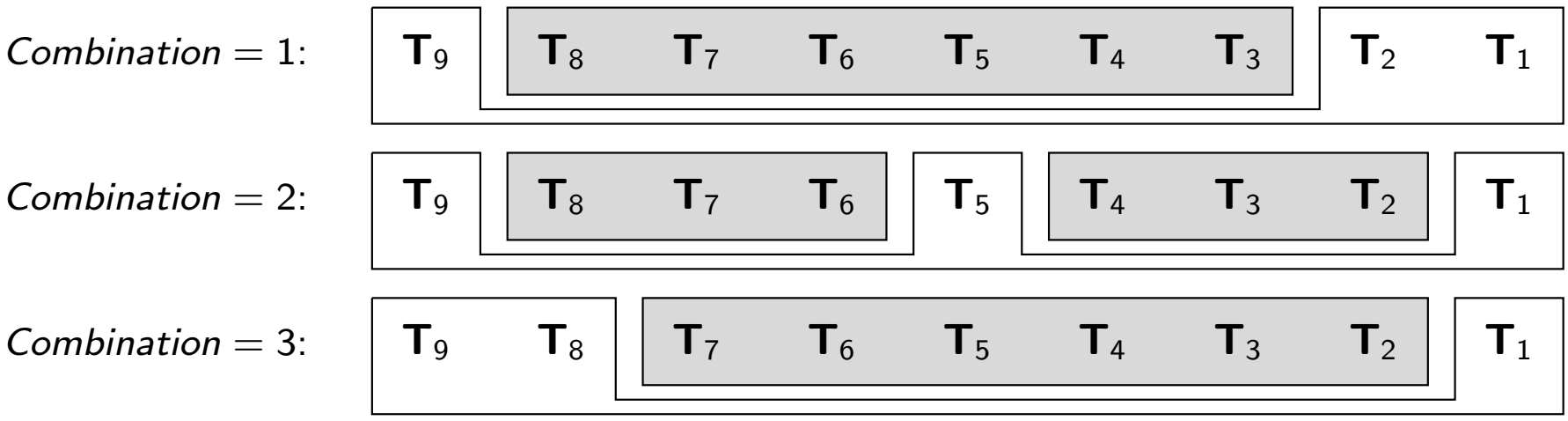
Dynamic Programming Solution

For example, with $V = 3$ and $Q = 3$, to build $G(1, 9)$:



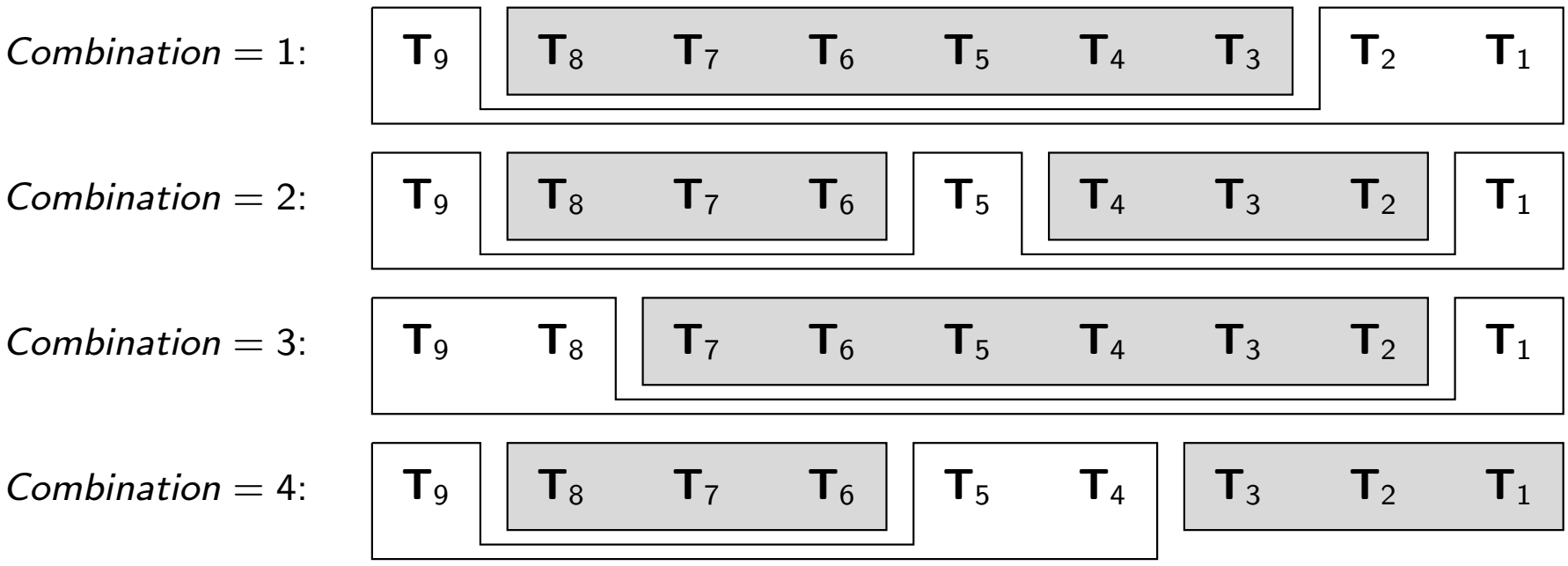
Dynamic Programming Solution

For example, with $V = 3$ and $Q = 3$, to build $G(1, 9)$:



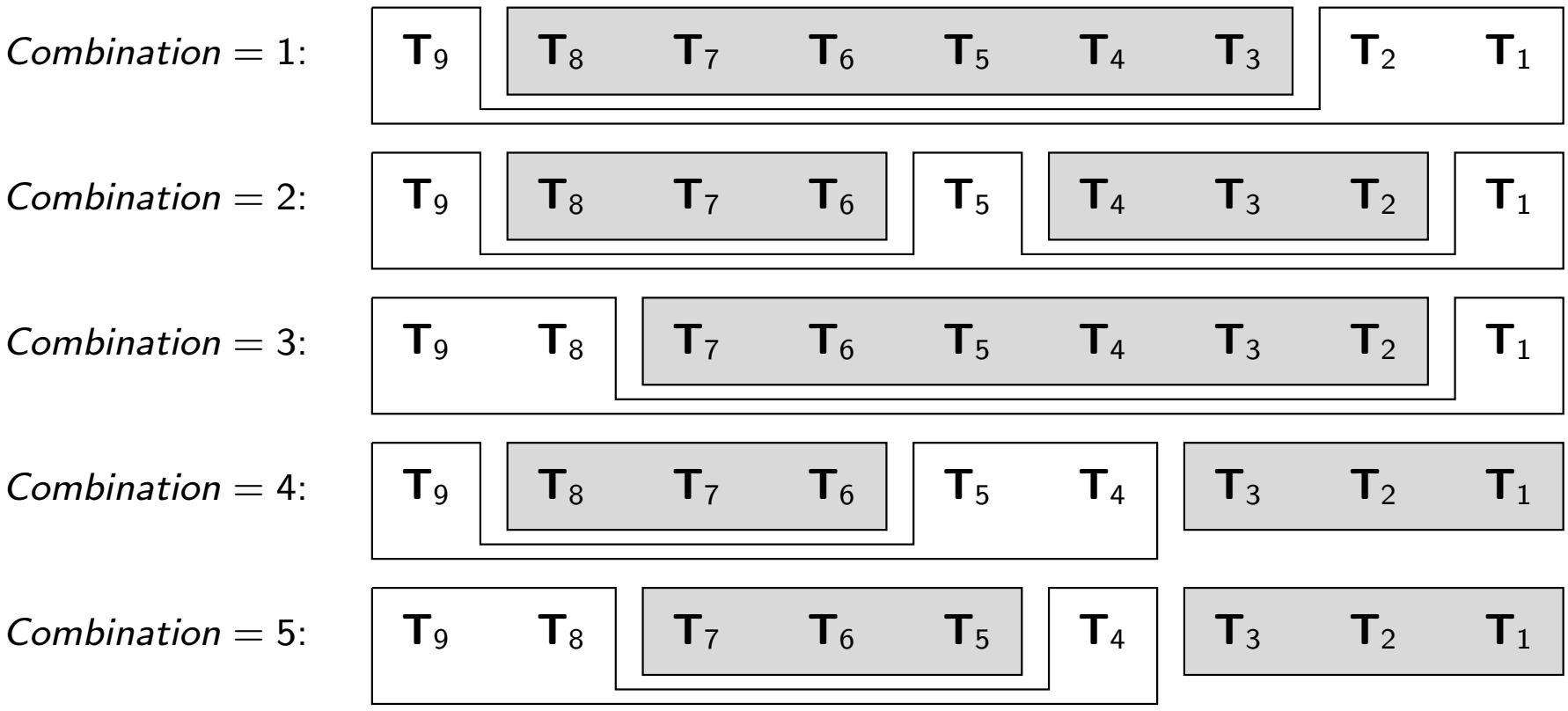
Dynamic Programming Solution

For example, with $V = 3$ and $Q = 3$, to build $G(1, 9)$:



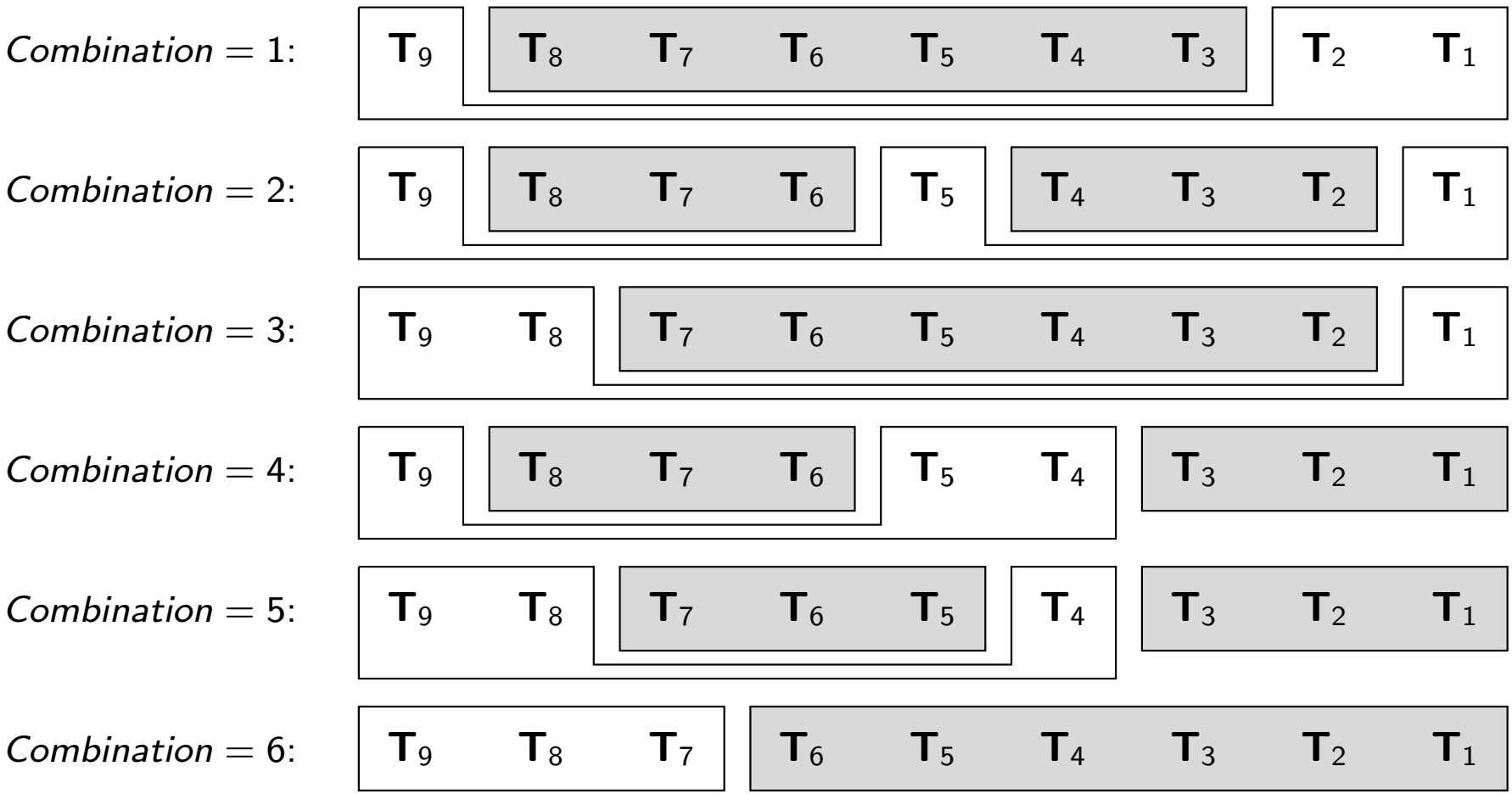
Dynamic Programming Solution

For example, with $V = 3$ and $Q = 3$, to build $G(1, 9)$:



Dynamic Programming Solution

For example, with $V = 3$ and $Q = 3$, to build $G(1, 9)$:



DYVIA: Experimental Evaluation on SCC

Different Benchmark Utilizations	Simulated Energy Consumption [Ws]				Measured Energy Consumption [Ws]			
	CCH	BUH	EOH	DYVIA	CCH	BUH	EOH	DYVIA
1	609.1	609.1	609.1	609.1	658.2	660.3	661.5	657.4
2	825.1	825.1	813.4	789.0	850.2	857.4	856.2	827.7
3	1041.7	1008.2	1022.5	949.7	1040.4	1009.5	1029.0	961.1
4	1039.6	1039.6	1023.0	948.1	1039.4	1046.8	1038.0	964.9
5	1041.9	1041.9	1025.0	983.5	1044.2	1042.2	1023.4	997.4
6	1369.9	1369.9	1369.9	1318.7	1335.4	1323.3	1332.2	1285.1
7	1487.7	1487.7	1486.9	1486.4	1455.5	1428.2	1465.1	1455.3
8	2378.0	2378.0	2326.7	2200.4	2317.5	2313.4	2124.3	2109.7
9	1349.3	1349.3	1349.3	1296.7	1314.7	1302.8	1327.2	1276.0
10	2324.3	2823.4	2324.3	2186.4	2273.2	2740.2	2267.4	2120.1
11	715.9	715.9	632.1	632.1	748.4	756.3	678.1	694.1
12	805.0	805.0	692.7	692.65	825.0	820.1	723.8	736.4

Algorithm	Simulated Energy Ratio			Measured Energy Ratio		
	Minimum	Average	Maximum	Minimum	Average	Maximum
CCH	1	1.0681	1.1622	1.0002	1.0562	1.1203
BUH	1	1.0842	1.2913	0.9814	1.0704	1.2925
EOH	1	1.0357	1.0789	0.9769	1.0278	1.0758

Simulations - Energy

- Simulations of energy for $V = \{2, 4, 6\}$ and $Q = \{2, 4, 6, 8\}$.
- The vertical line shows the range of the energy consumption ratios.
- The bar represents the average energy consumption ratio.