Energy-aware Scheduling for Frame-based Tasks on Heterogeneous Multiprocessor Platforms

Dawei Li and Jie Wu

Temple University, Philadelphia, PA, USA

Outline

1. Introduction

2. System Model

3. Relaxation-based Iterative Rounding Algorithm

4. Simulation

5. Conclusion

Introduction

- Dynamic Voltage and Frequency Scaling (DVFS)
- Motivational Example

Dynamic Voltage and Frequency Scaling (DVFS)

- Background:
  - high performance lead to high energy consumption.

- DVFS allows processors to adjust
  - the supply voltage or
  - the execution frequency

  to operate on different power/energy levels.
- It is considered an effective way to achieve the goal of saving energy

Motivational Example

- Assumptions:
  - Processors must execute at a **same frequency**, and after initial setting the execution **frequency cannot change**.
  - **Power consumption**: $p = f^3$, f can be from [0, infinity). When a processor is **idle**, it consumes **no power**.
  - Tasks **cannot be preempted**.

- Denote when executing at a same fixed frequency f_s, the execution time of task i on the jth processor M_j, by $t_{i,j}$.

- Consider scheduling **four** tasks on **two** processors.

$t_{1,1} = 30$; $t_{1,2} = 50$; $t_{2,1} = 12$; $t_{2,2} = 35$;
$t_{3,1} = 15$; $t_{3,2} = 24$; $t_{4,1} = 12$; $t_{4,2} = 10$.
Deadline $D = 100$.

Motivational Example

- Goal:
  - minimize the overall energy consumption and no task misses the deadline.
- Two steps:
  - Partition the tasks to processors.
  - Set the same frequency for the processors such that the processor with the highest workload finishes its task(s) exactly at the deadline D.

Motivational Example

o Existing partitioning approaches:

o Min-min:

In the first phase, the set of tasks' **minimum** expected completion times is calculated (for all unassigned tasks).

In the second phase, the task with the over-all **minimum** expected completion time in the set (derived by the first phase) is chosen and assigned to the corresponding processor.

Then, this task is removed from the unassigned task set, and the procedure is repeated until all tasks are assigned.

oMax-min:

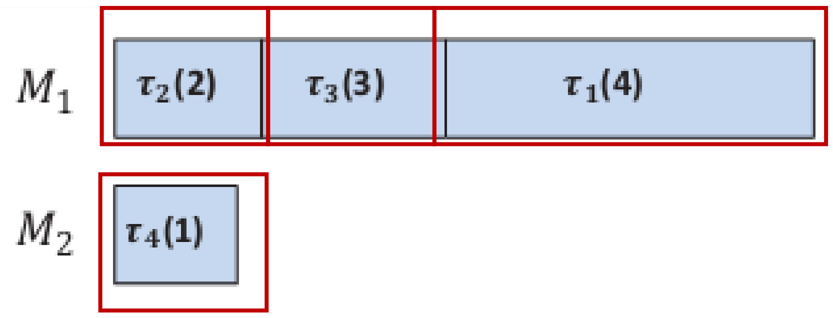
This heuristic is very similar to the min-min heuristic.

The only difference is that, in the second phase, the task with the overall **maximum** expected completion time among all of the unassigned tasks is chosen and assigned to the corresponding processor.

Motivational Example

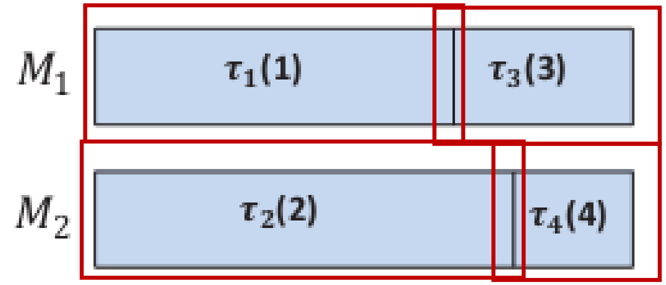
$$t_{1,1} = 30;\ t_{1,2} = 50;\ t_{2,1} = 12;\ t_{2,2} = 35;$$
$$t_{3,1} = 15;\ t_{3,2} = 24;\ t_{4,1} = 12;\ t_{4,2} = 10;$$

- Partition by min-min
  - Min(30, 12, 15, 10)



(a) Partitioned by the min-min heuristic

- Partition by max-min
  - Max(30, 12, 15, 10)



(b) Partitioned by the max-min heuristic

- These two approaches are commonly Used to achieve workload balanced partition. Does a workload balanced partition really result in less energy consumption?

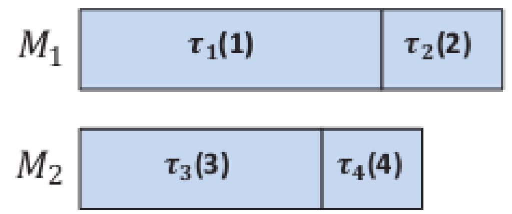
Motivational Example

- What is the **ideal optimal solution**, if we allow splitting the tasks arbitrarily?
  - 90% of τ_1 on processor M_1, 10% on processor M_2.
  - 100% of τ_2 on processor M_1.
  - 100% of τ_3 on processor M_2.
  - 100% of τ_4 on processor M_2.

- Intuition: a solution which is *close to* the **ideal optimal solution** should be better in terms of minimizing energy consumption. Two possible methods to get a better solution:
  - Relaxation-based Naive Rounding Algorithm (RNRA) (Does not work well for general cases)
  - Relaxation-based Iterative Rounding Algorithm (RIRA) (The method we propose).

Motivational Example

- Partition by RIRA



(c) Partitioned by our RIRA approach

- After partitioning, set the frequency such that the processor with the highest workload finishes its task(s) exactly at the deadline D.

- Energy consumption comparisons:

Overall energy consumption		
min-min	max-min	Our Proposed RIRA
$21.7683 f_s^3$	$18.225 f_s^3$	$13.4064 f_s^3$

- Our proposed RIRA achieves a much better performance in terms of minimizing energy consumption.

Outline

1. Introduction

2. System Model

3. Relaxation-based Iterative Rounding Algorithm

4. Simulation

5. Conclusion

System Model

- Task Model
- Platform Model
- Problem Definition

Task Model

- A set of frame-based tasks $\{\tau_1, \tau_2, \ldots, \tau_n\}$ that are released at time 0 and share the same deadline D.

- Each task τ_i's execution requirement: C_i.

- The execution time of task τ_i when it is executed at frequency f on a processor with execution efficiency λ:

$$C_i/(\lambda f).$$

- Tasks cannot be preempted.

Platform Models

- Define $\lambda_{i,j}$ as the execution efficiency of processor M_j when it is processing task τ_i. Then the execution time of task τ_i, if it is executed on processor M_j, can be calculated as $C_i/(\lambda_{i,j} f)$.

- Power consumption models
  - Processors can operate in two modes.
    - Run mode: $p = f^3$
    - Idle mode: consumes zero power
  - When a processor has no task to execute, it transitions into idle mode immediately without any overhead.

Platform Models

- Three platform types
  - Dependent Platforms without Runtime Adjusting
    all of the processors must operate at a same frequency, and the same execution frequency cannot be adjusted during runtime.

  - Dependent Platforms with Runtime Adjusting
    all the processors must operate at a shared frequency, and the shared frequency can be adjusted during runtime.

  - Independent Platforms
    processors can operate at different frequencies at any time, and can adjust their execution frequencies independently

Problem Definition

- Given a set of frame-based tasks, our goal is to schedule all of the tasks on m heterogeneous processor, such that no task misses the deadline and the overall energy consumption is minimized.

- A scheduling consists of two steps.
  - The first and the main step is to produce a partition with the goal of achieving minimal energy consumption.
  - Frequency setting for different platform types.

Problem Definition

- For dependent platforms without runtime adjusting
  - The shared frequency should be chosen such that the processor with the greatest workload completes all of the tasks assigned to it exactly at the deadline D.
- For dependent platforms with runtime adjusting
  - We can further determine the optimal frequencies, for the running processors, in different time intervals.
- For independent platforms
  - All processors are slowed down independently such that each processor completes all of the tasks assigned to it exactly at deadline D.

Outline

1. Introduction

2. System Model

3. Relaxation-based Iterative Rounding Algorithm

4. Simulation

5. Conclusion

Relaxation-based Iterative Rounding Algorithm (RIRA)

- Scheduling on Dependent Platforms without Runtime Adjusting
- Scheduling on Dependent Platforms with Runtime Adjusting
- Scheduling on Independent Platforms

Scheduling on Dependent Platforms without Runtime Adjusting

- Intuition:
  - the task with the greatest execution requirement can be considered as the most "influential" task, and thus, should be considered first.

- Define the Average Execution Cycles (AEC) of task τ_i as $AEC_i = \frac{1}{m}\sum_{j=1}^{m} c_i/\lambda_{i,j}$ and sort the tasks in the order $\tau_{i_1}, \tau_{i_2}, \dots, \tau_{i_n}$, such that $AEC_{i_1} \geq AEC_{i_2} \geq \cdots \geq AEC_{i_n}$.
  - This is also the order that we will assign tasks in.
  - Without loss of generality, from now on, we assume that tasks are already in the order: $AEC_1 \geq AEC_2 \geq \cdots \geq AEC_n$

Scheduling on Dependent Platforms without Runtime Adjusting

- Formulation: to achieve a partition with the objective of saving energy, the problem can be formulated as:

$$
\begin{aligned}
min \quad & E_{total} = f^2 \textstyle\sum_{j=1}^{m} \left(\sum_{i=1}^{n} \frac{x_{i,j} C_i}{\lambda_{i,j}} \right) \\
s.t. \quad & \textstyle\sum_{i=1}^{n} \frac{x_{i,j} C_i}{\lambda_{i,j}} - fD \leq 0, \forall j = 1, 2, \cdots, m. \\
& \boxed{x_{i,j} = 0 \ or \ 1,} \forall i = 1, 2, \cdots, n; j = 1, 2, \cdots, m. \\
& \textstyle\sum_{j=1}^{m} x_{i,j} = 1, \forall i = 1, 2, \cdots, n. \\
& f \geq 0.
\end{aligned}
$$

$$\boxed{0 \leq x_{i,j} \leq 1,}$$

- This binary integer programming problem is NP-hard.
- Relaxation: (allow splitting the tasks arbitrarily to achieve the ideal optimal solution.)

Scheduling on Dependent Platforms without Runtime Adjusting

- Rounding:
  - since tasks are already in our desired order, the solutions $x_{1,1}, x_{1,2}, \ldots, x_{1,m}$ indicate the optimal assignment of the most influential task τ_1.
  - We find the maximum among $x_{1,1}, x_{1,2}, \ldots, x_{1,m}$, denoted by x_{1,j^*}, and assign τ_1 to processor M_{j^*}.
  - Which means: $x_{1,j} = 0$, for $j \neq j^*$ and $x_{1,j^*} = 1$.

- Update the problem:

$$
\begin{aligned}
min \quad & E_{total} = f^2 \textstyle\sum_{i=1}^{m} (\sum_{i=1}^{n} \frac{x_{i,j} C_i}{\lambda_{i,j}}) \\
s.t. \quad & \textstyle\sum_{i=1}^{n} \frac{x_{i,j} C_i}{\lambda_{i,j}} - fD \leq 0, \forall j = 1, 2, \cdots, m. \\
& 0 \leq x_{i,j} \leq 1, \forall i = 2, \cdots, n; \forall j = 1, 2, \cdots, m. \\
& \textstyle\sum_{j=1}^{m} x_{i,j} = 1, \forall i = 2, \cdots, n \\
& f \geq 0.
\end{aligned}
$$

Scheduling on Dependent Platforms without Runtime Adjusting

Overall algorithm: Iteratively round and update until $(n-1)$ tasks have been assigned. For the last task n, we just select the assignment that achieves the minimal overall energy consumption among all possible assignments of the last task.

Algorithm 2 RIRA

Input:

The sorted task set $\mathcal{T} = \{\tau_1, \tau_2, \cdots, \tau_n\}$ and associated WCECs, $C_1, C_2, \cdots, C_n$; processor efficiency matrix, $\lambda_{n\times m}$;

Output:

Binary matrix $Assign_{n\times m}$ indicating the final assignment;

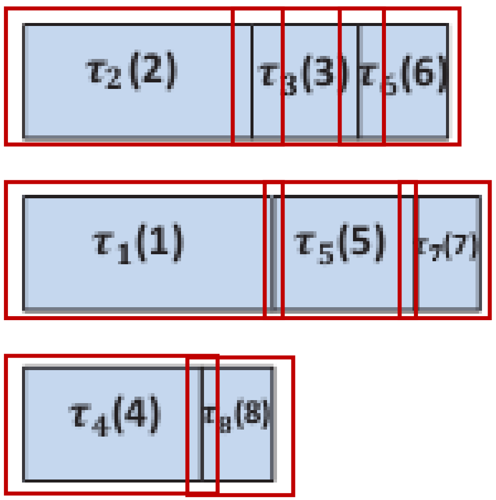
Initialize the the assignment matrix: $Assign_{i,j} = 0, \forall i = 1, 2, \cdots, n; j = 1, 2, \cdots, m$;
for $i := 1$ *to* $n-1$ **do**
  Solve Optimization problem P_i;
  $x_{i,j^*} = max(x_{i,1}, x_{i,2}, \cdots, x_{i,m})$;
  for $j := 1$ *to* m **do**
    $x_{i,j} = 0$;
  end for
  $Assign_{i,j^*} = 1$;
  $x_{i,j^*} = 1$;
  Update the optimization problem to be P_{i+1};
end for
Assign the last task τ_n such that the final assignment achieves the minimal energy consumption among all possible assignments for the last task. Denote this by $Assign_{n,j'} = 1$;
return $Assign$;

Scheduling on Dependent Platforms without Runtime Adjusting

- Example:

i	C_i	$\lambda_{i,j}$			$t_{i,j}$		
		$j=1$	$j=2$	$j=3$	$j=1$	$j=2$	$j=3$
1	7	.7	.4	.1	10	17.5	70
2	8	.5	.2	.3	16	40	26.67
3	3	.4	.1	.2	7.5	30	15
4	5	.5	.2	.4	10	25	12.5
5	9	.6	.9	.7	15	10	12.86
6	5	.8	.3	.5	6.25	16.67	10
7	4	.3	.9	.6	13.33	4.44	6.67
8	4	.4	.6	.8	10	6.67	5

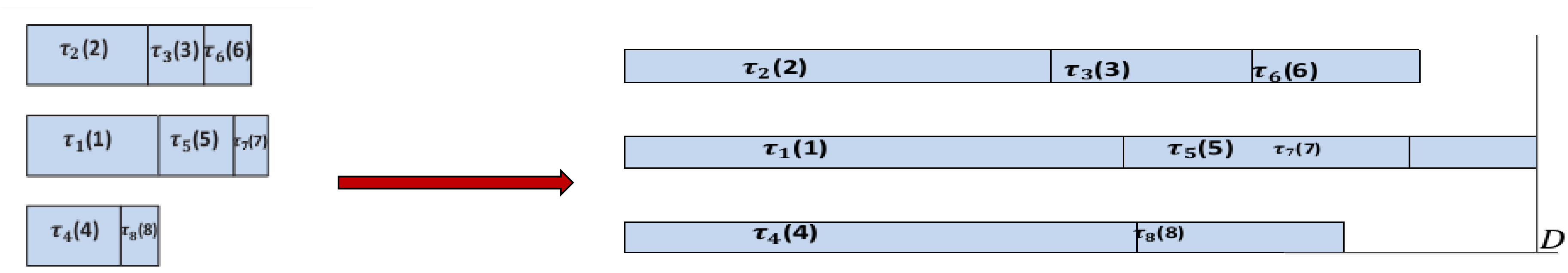
i	Relaxed Assignment $x_{i,j}$ for P_i			$Assign_{8\times 3}$		
	$j=1$	$j=2$	$j=3$	$j=1$	$j=2$	$j=3$
1	0.2920	0.7080	0	0	1	0
2	1	0	0	1	0	0
3	0.99984	0.00001	0.00015	1	0	0
4	0.00013	0.00001	0.99986	0	0	1
5	0	0.5379	0.4621	0	1	0
6	0.6504	0	0.3496	1	0	0
7	0	0.5062	04938	0	1	0
8	-	-	-	0	0	1



(d) RIRA

Scheduling on Dependent Platforms without Runtime Adjusting

- After Partition
  - The shared frequency is set such that the processor with the greatest workload finishes its tasks exactly at the deadline.



(d) RIRA

- Energy Consumption Comparisons:
  - min-min heuristic 11.3
  - max-min heuristic 10.7
  - RNRA 8.464
  - RIRA 8.08

Scheduling on Dependent Platforms with Runtime Adjusting

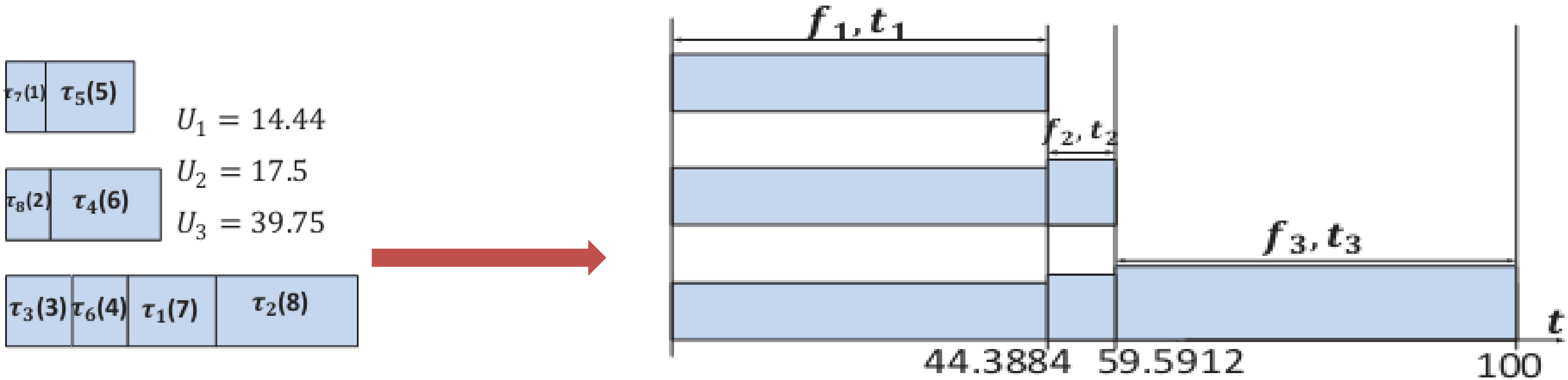
- Adopt the same partition as that for Dependent Platforms without Runtime Adjusting.
- After partition.
  - We can determine the optimal frequency setting for each time interval.

$$U_j = \sum_{i=1}^{n} \frac{x_{i,j} C_i}{\lambda_{i,j}}$$ (sort in ascending order)

$$\begin{aligned} min \quad & E_{total} = \sum_{j=1}^{m} (m-j+1) f_j^2 (U_j - U_{j-1}) \\ s.t. \quad & \sum_{j=1}^{m} \frac{U_j - U_{j-1}}{f_j} \le D \\ & 0 \le \frac{U_j - U_{j-1}}{f_j} \le D, f_j \ge 0, \forall j = 1, 2, \cdots, m. \end{aligned}$$

$$f_j = \frac{\sum_{j=1}^{m} (U_j - U_{j-1}) \sqrt[3]{m-j+1}}{D \sqrt[3]{m-j+1}}$$

Scheduling on Dependent Platforms with Runtime Adjusting



(a) Sorted workloads after the min-min partition

(b) Runtime frequency adjusting, $f_1 = 0.3254$, $f_2 = 0.3725$, $f_3 = 0.4693$, $t_1 = 44.3884$, $t_2 = 8.2028$, $t_3 = 47.4088$

- Energy Consumption Comparisons
  - min-min heuristic 10.3375,
  - max-min heuristic 10.4740,
  - RNRA 8.1617,
  - RIRA 7.8776.

Scheduling on Independent Platforms

- Problem formulation and relaxation

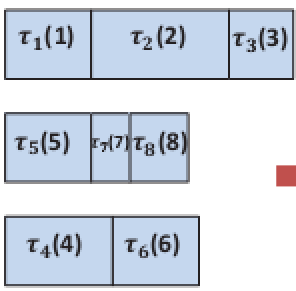
$$
\begin{aligned}
min \quad & E_{total} = \frac{1}{D^2}\sum_{j=1}^{m}\left(\sum_{i=1}^{n}\frac{x_{i,j}C_i}{\lambda_{i,j}}\right)^3 \\
s.t. \quad & \sum_{j=1}^{m} x_{i,j} = 1, \forall i = 1, 2, \cdots, n \\
& 0 \leq x_{i,j} \leq 1, \forall i = 1, 2, \cdots, n, \forall j = 1, 2, \cdots, m.
\end{aligned}
$$

- Assign the most influential task, then update the problem.

$$
\begin{aligned}
min \quad & E_{total} = \frac{1}{D^2}\sum_{j=1}^{m}\left(\sum_{i=1}^{n}\frac{x_{i,j}C_i}{\lambda_{i,j}}\right)^3 \\
s.t. \quad & \sum_{j=1}^{m} x_{i,j} = 1, \forall i = 2, \cdots, n \\
& 0 \leq x_{i,j} \leq 1, \forall i = 2, \cdots, n; \forall j = 1, 2, \cdots, m.
\end{aligned}
$$

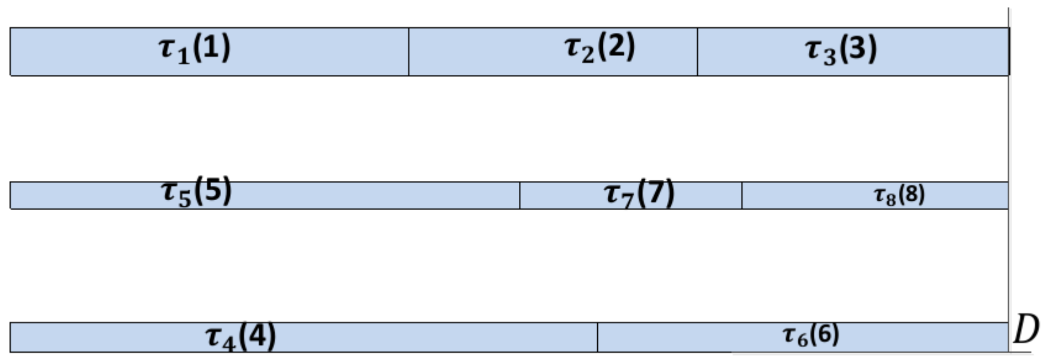
Scheduling on Independent Platforms

- Partition by RIRA.



(b) RIRA

- After partition, processors adjust their execution frequencies independently such that each processor finishes its tasks exactly at the deadline D.



- Energy consumption comparisons
  - min-min heuristic 7.11
  - max-min heuristic 8.92
  - RNRA: 6.14.
  - RIRA: 5.84.

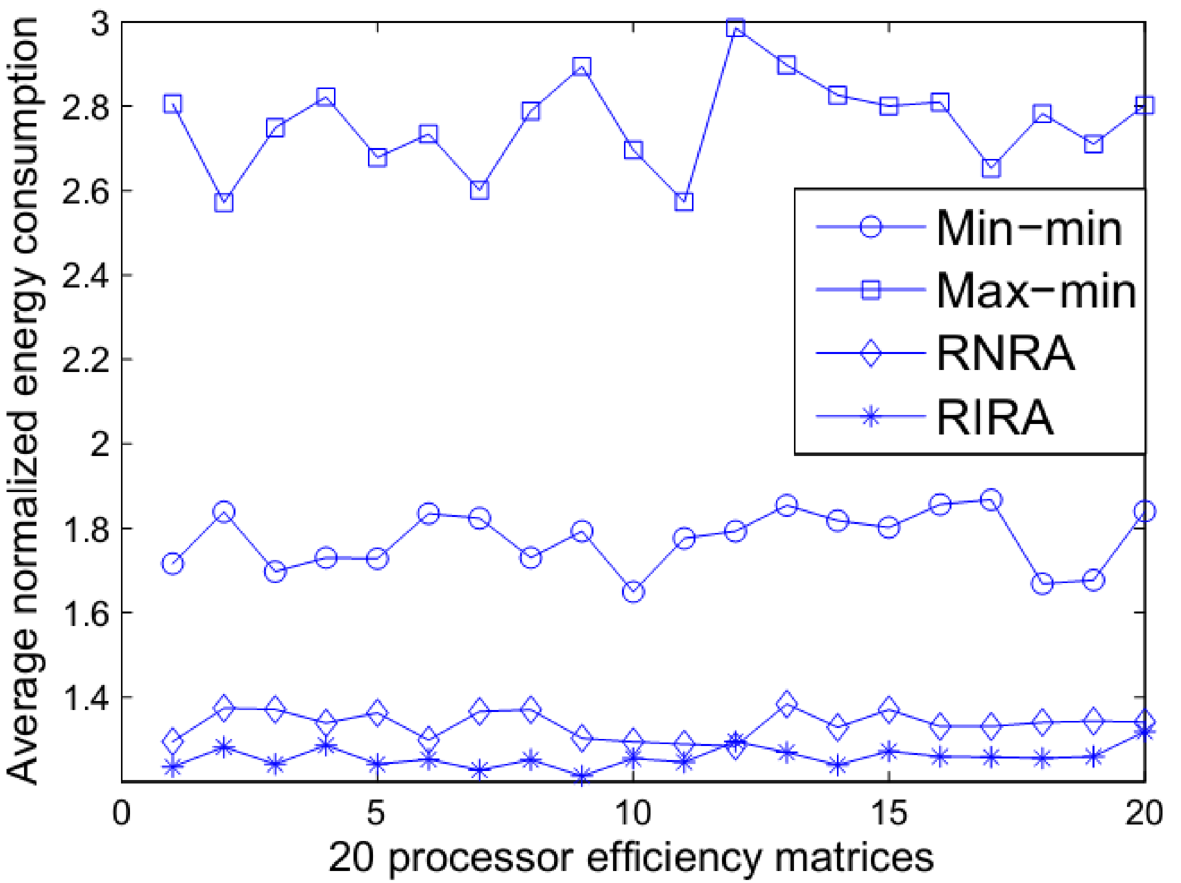
Outline

1. Introduction
2. System Model
3. Relaxation-based Iterative Rounding Algorithm
4. Simulations
5. Conclusion

Simulations

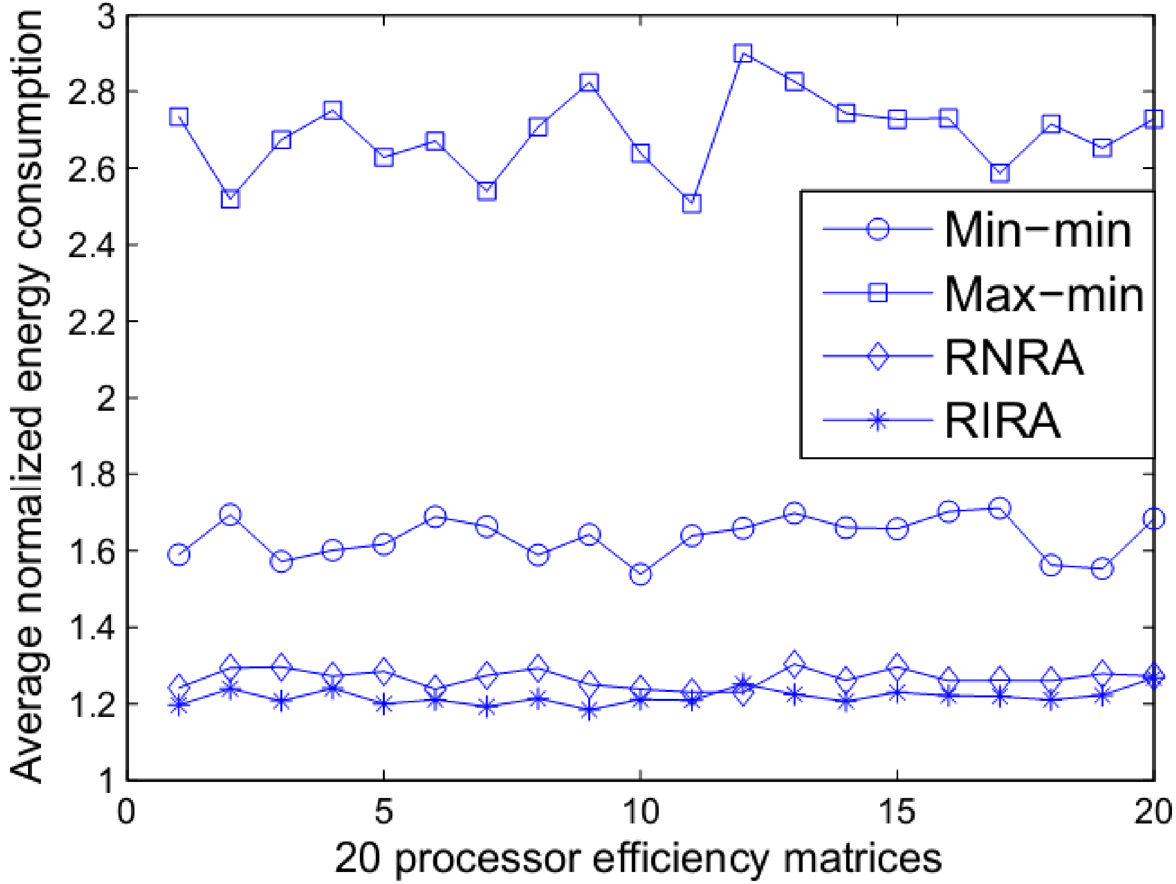
- Settings:
  - Scheduling 20 tasks to 6 processors.
  - For each platform type, we randomly generate 20 processor efficiency matrix.
  - For each efficiency matrix, we randomly generate 20 sets of task execution.
  - On a given type of platform, for a given processor efficiency matrix, we compare the average normalized energy consumption (normalized by the ideal optimal solution) of the 20 randomly generated tasks.

Simulations



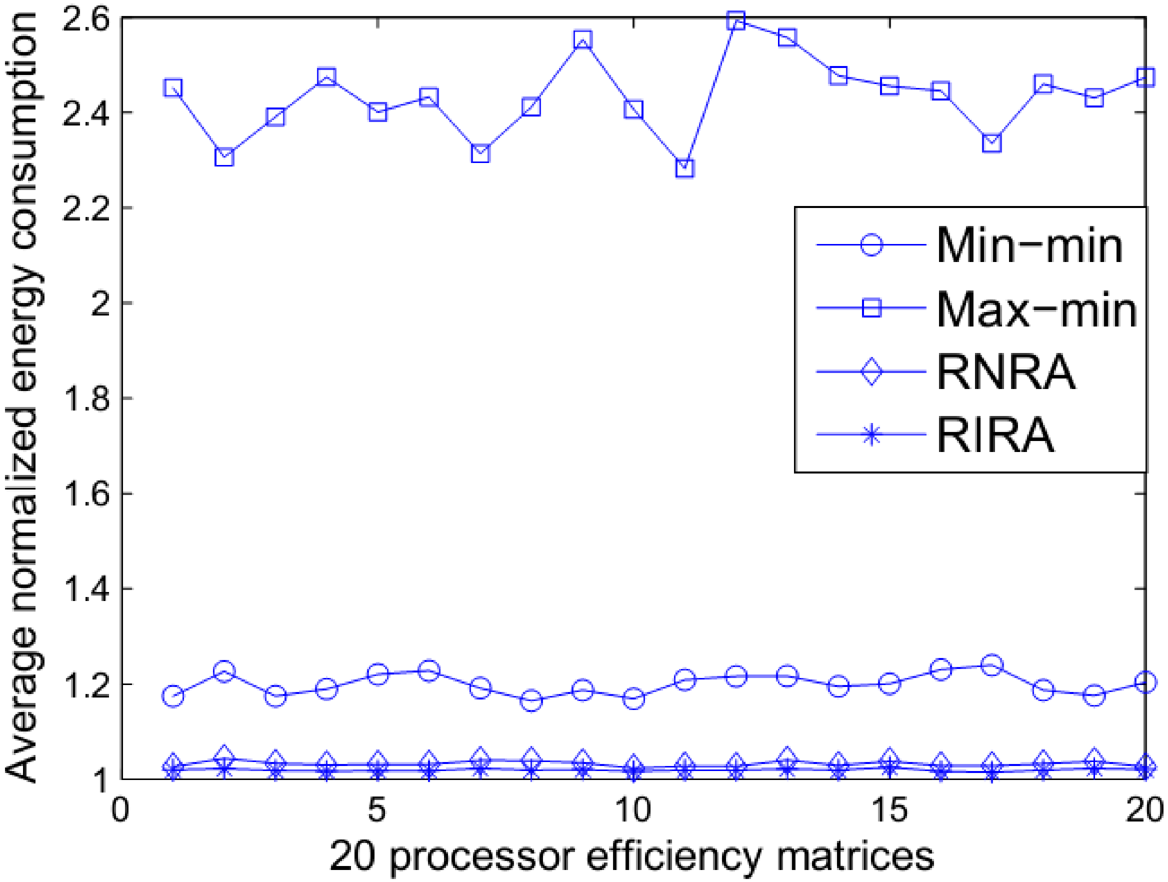
(a) Dependent Platform without Runtime Adjusting

Simulations



(b) Dependent Platform with Runtime Adjusting

Simulations



(c) Independent Platform

Outline

1. Introduction
2. System Model
3. Relaxation-based Iterative Rounding Algorithm
4. Simulations
5. Conclusion

Conclusion

o A Relaxation-based Iterative Rounding Algorithm (RIRA) is proposed for energy-aware task allocating and scheduling.

oThree typical platform types are considered. Our proposed RIRA is suitable for all of the three platform types.

oExperiments and simulations verify the strength of our approach.

Thank you!

Questions?

dawei.li@temple.edu