INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING
ICPP / 2021 / CHICAGO / USA
acm In-Cooperation
sighpc
AUGUST 9-12, 2021

# Multi-Resource List Scheduling of Moldable Parallel Jobs under Precedence Constraints

Lucas Perotin (speaker)[1] Hongyang Sun[2]
Padma Raghavan[2]

[1]École Normale Supérieure de Lyon, France

[2]Vanderbilt University, USA

ENS DE LYON

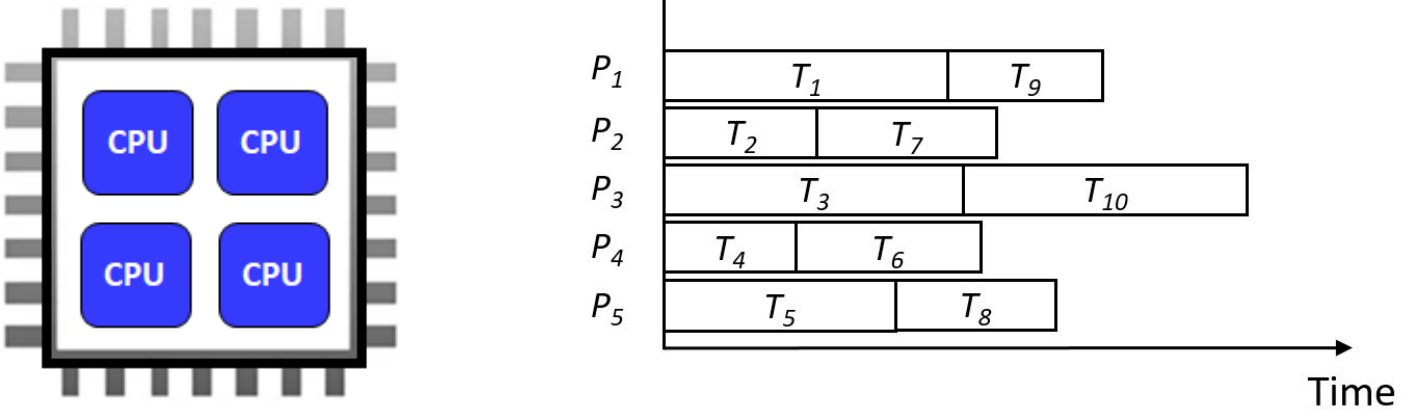
VANDERBILT UNIVERSITY

August 10, 2021

# Introduction

## Single-resource scheduling

- Most traditional scheduling problems target a single type of resource (e.g., CPUs)

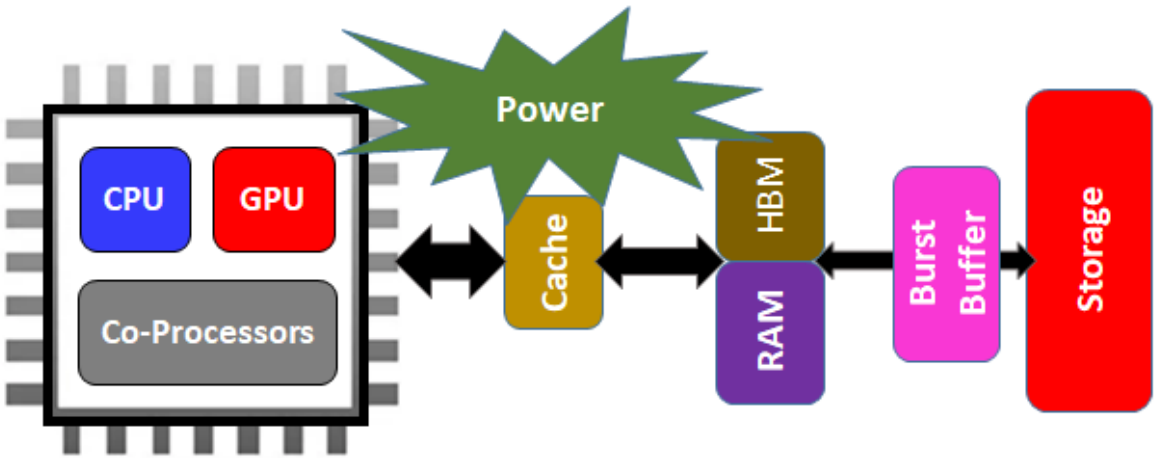



- For example: classic NP-complete problem of makespan minimization for moldable tasks with precedence constraints on identical machines ($P|moldable, prec|C_{\max}$). Best known algorithm is 3.42-approx. [Chen 2018]

# Introduction

**The case for multi-resource scheduling**

- HPC systems embrace more heterogeneous components (e.g., CPU, GPU, FPGA, MIC, APU)
- Data-intensive applications drive architecture enhancement for better data-transfer efficiency (e.g., High-Bandwidth Memory, Partitionable Cache, Burst Buffers)



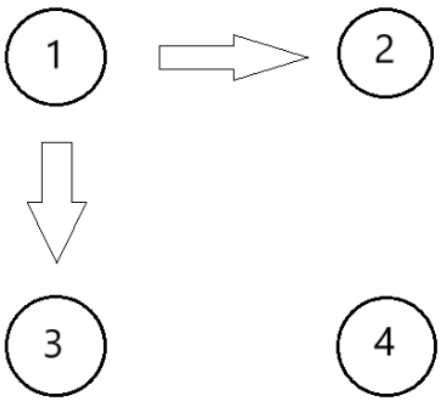

To achieve optimal system/application performance, multiple types of resources (e.g., CPU, GPU, memory, cache, I/O) should be scheduled simultaneously

# A Simple Example

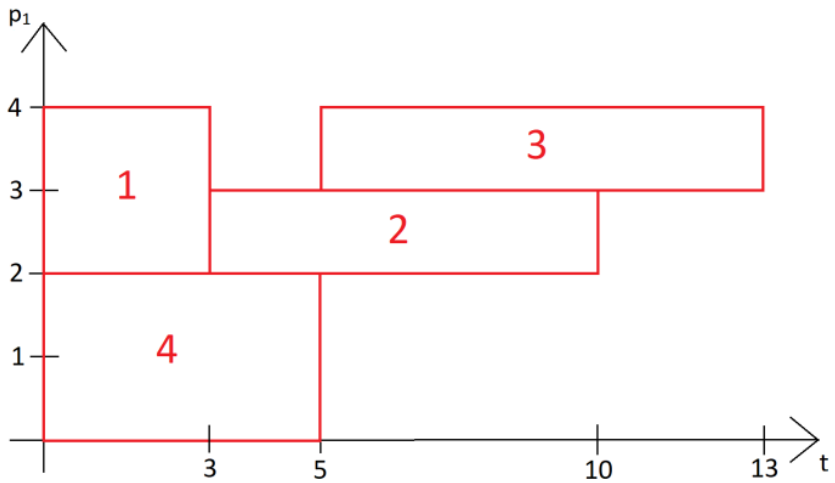
Resource 1 has 4 units.
Resource 2 has 6 units.

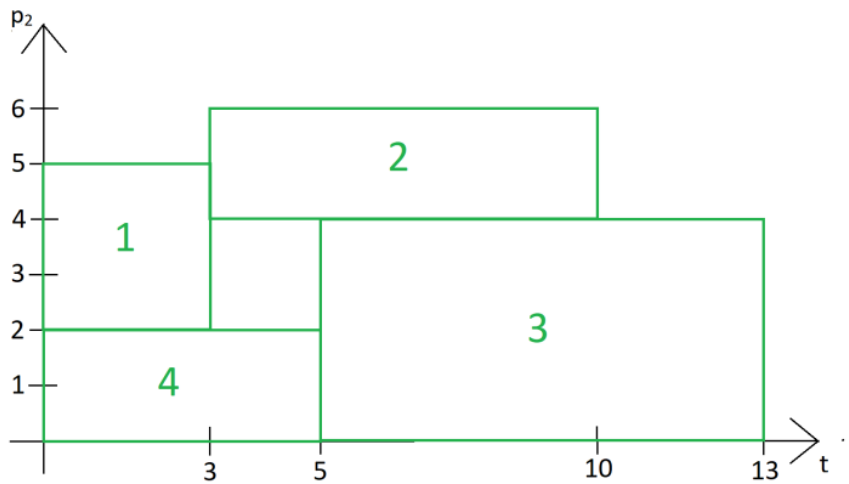
| Tasks | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Time | 3 | 7 | 8 | 5 |
| Resource 1 | 2 | 1 | 1 | 2 |
| Resource 2 | 3 | 2 | 4 | 2 |

The precedence constraints are :




Possible realisation :



# Scheduling Model

**A multi-resource scheduling model:**

- System with $d$ resource types; $i$-th type has $P^{(i)}$ identical resources
- Set $\{1, 2, \cdots, n\}$ of moldable tasks released at time 0
- There may be precedence constraints between tasks
- Each task $j$'s execution time $t_j(\vec{p}_j)$ depends on its resource allocation vector $\vec{p}_j = (p_j^{(1)}, p_j^{(2)}, \cdots, p_j^{(d)})$ and is known
- Assumptions: *non-increasing execution time, non-superlinear speedup* with respect to any ressource type.

$$\vec{p}_j \preceq \vec{q}_j \ \left(\text{or } p_j^{(i)} \leq q_j^{(i)}, \forall i\right) \implies t_j(\vec{p}_j) \geq t_j(\vec{q}_j),$$

$$\implies t_j(p_j) \leq \left(\max_{i=1\ldots d} \frac{q_j^{(i)}}{p_j^{(i)}}\right) \cdot t_j(q_j) \ .$$

# Scheduling Objective

**Scheduling objective:**
Find a moldable schedule, i.e., resource allocation vector $\vec{p}_j$ and starting time $s_j$ for each task $j$ which

- minimizes makespan: $T = \max_j(s_j + t_j(\vec{p}_j))$
- subject to resource constraint: $\sum_{j \text{ active at time } t} p_j^{(i)} \leq P^{(i)}, \forall i, t$
- subject to precedence constraint: $j_1 \to j_2 \implies s_{j_2} \geq s_{j_1} + t_{j_1}$

**Approximation ratio:**
An algorithm is said to be *r-approximation* if its makespan satisfies $\frac{T}{T_{\text{OPT}}} \leq r$ for any task graph, where $T_{\text{OPT}}$ denotes the optimal makespan.

# Main Results

Approximation ratios that increase linearly with number $d$ of resource types for different task graphs

- For general task graphs:
  - $1.619d + 2.545\sqrt{d} + 1$ for all $d \geq 1$
  - $d + 3\sqrt[3]{d^2} + O(\sqrt[3]{d})$ for $d \geq 22$.
- For independant tasks (previous best ratio $= 2d$ [Sun et al. 18]):
  - $1.619d + 1$ for all $d \geq 1$
  - $d + 2\sqrt{d-1}$ for $d \geq 4$.

Lower bound of $d$ on the approximation ratio of any deterministic list-based scheduling algorithm with local job priority consideration.

# Outline

# Preliminaries

**Definitions**: for a given resource allocation $\mathbf{p} = (\vec{p}_1, \vec{p}_2, \cdots, \vec{p}_n)^T$

- Total task area (normalized): $A(\mathbf{p}) = \sum_{j=1}^{n} \sum_{i=1}^{d} \frac{p_j^{(i)}}{P^{(i)}} \cdot t_j(\vec{p}_j)$
- Critical-Path: $C(\mathbf{p}) = \max_f \sum_{j \in f} t_j(\vec{p}_j)$ over all paths $f$ in the graph

Analogous to *Area bound* ($T_1/P$) and *Critical-Path bound* ($C_{max}$) in single-resource scheduling

# Preliminaries

**Definitions**: for a given resource allocation $\mathbf{p} = (\vec{p}_1, \vec{p}_2, \cdots, \vec{p}_n)^T$

- Total task area (normalized): $A(\mathbf{p}) = \sum_{j=1}^{n} \sum_{i=1}^{d} \frac{p_j^{(i)}}{P^{(i)}} \cdot t_j(\vec{p}_j)$
- Critical-Path: $C(\mathbf{p}) = \max_f \sum_{j \in f} t_j(\vec{p}_j)$ over all paths $f$ in the graph

Analogous to *Area bound* ($T_1/P$) and *Critical-Path bound* ($C_{max}$) in single-resource scheduling

**Lower bound** (on makespan): $L(\mathbf{p}, d) = \max\left(\frac{A(\mathbf{p})}{d}, C(\mathbf{p})\right)$

## Proposition

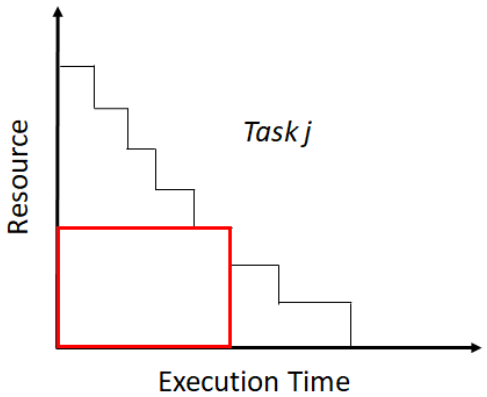
*The **optimal makespan** satisfies*

$$T_{\text{OPT}} \geq L_{\min}(d) = \min_{\mathbf{p}} L(\mathbf{p}, d)$$

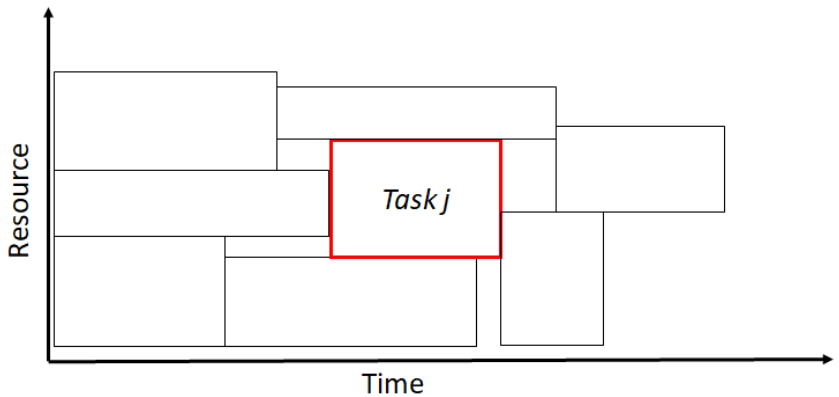
# Moldable Scheduling

**Two-phase approach** [Turek et al. 92]:

- *Phase 1*: Determines a resource allocation for each moldable task




- *Phase 2*: Constructs a rigid schedule based on the fixed resource allocations of all tasks



# Phase 1: Resource Allocation

- Step (1): *Filter possible allocation*. For each task $j$:
  - Linearize all $P = \prod_{i=1}^{d}(P^{(i)} + 1)$ allocations
  - Remove ones with both higher execution time and larger area

# Phase 1: Resource Allocation

- Step (1): *Filter possible allocation*. For each task $j$:
  - Linearize all $P = \prod_{i=1}^{d}(P^{(i)} + 1)$ allocations
  - Remove ones with both higher execution time and larger area
- Step (2): *Find best tradeoff*. For all $n$ tasks:
  - Use linear programming to obtain $L(\mathbf{p}^*, d) = L^*_{\min}(d)$ (Adapted from the Discrete Time-Cost Tradeoff problem [Skutella 98])
  - For a given $\rho \in (0, 1)$, round to $\mathbf{p}'$ such that

$$C(\mathbf{p}') \leq \frac{T_{\text{OPT}}}{\rho} \text{ and } A(\mathbf{p}') \leq \frac{T_{\text{OPT}}}{1-\rho}$$

# Phase 1: Resource Allocation

- Step (1): *Filter possible allocation*. For each task $j$:
  - Linearize all $P = \prod_{i=1}^{d}(P^{(i)} + 1)$ allocations
  - Remove ones with both higher execution time and larger area
- Step (2): *Find best tradeoff*. For all $n$ tasks:
  - Use linear programming to obtain $L(\mathbf{p}^*, d) = L^*_{\min}(d)$ (Adapted from the Discrete Time-Cost Tradeoff problem [Skutella 98])
  - For a given $\rho \in (0, 1)$, round to $\mathbf{p}'$ such that

$$C(\mathbf{p}') \leq \frac{T_{\text{OPT}}}{\rho} \text{ and } A(\mathbf{p}') \leq \frac{T_{\text{OPT}}}{1-\rho}$$

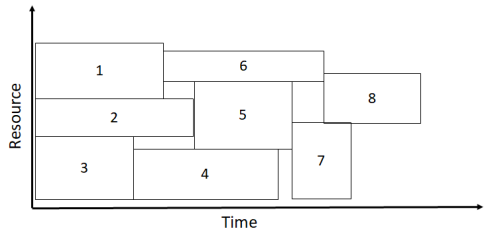
- Step (3): *Adjust allocation*. For each task $j$ and a given $\mu \in (0, 0.5)$:

$$\forall i, p_j^{(i)} = \begin{cases} \lceil \mu P^{(i)} \rceil, & \text{if } {p'}_j^{(i)} > \lceil \mu P^{(i)} \rceil \\ {p'}_j^{(i)}, & \text{otherwise.} \end{cases}$$

# Phase 2: Rigid Scheduling

**For a fixed resource allocation**:

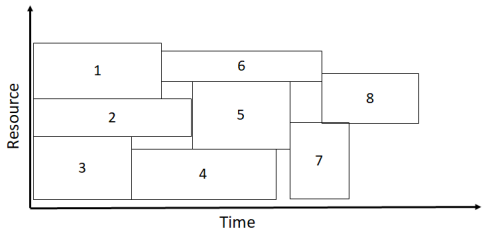
- List Scheduling:
  - Arrange all tasks in a list according to priority rules.
  - Whenever an existing task completes, scan the list and schedule first ready task that fits.



# Phase 2: Rigid Scheduling

**For a fixed resource allocation**:

- List Scheduling:
  - Arrange all tasks in a list according to priority rules.
  - Whenever an existing task completes, scan the list and schedule first ready task that fits.




## Proposition

*This two-phase algorithm with* $\mu = 1 - \frac{1}{\phi}$ *and* $\rho = \frac{1}{\sqrt{\phi d}+1}$ *satisfies:*

$$T \leq (\phi d + 2\sqrt{\phi d} + 1) \cdot T_{opt}$$

*where* $\phi = \frac{1+\sqrt{5}}{2} < 1.619$ *is the golden ratio*

# Outline

# Analytical Tools

**Definitions**: The processing time [0,T] is subdivided in three sets:

- $I_1$: For all $i$, less than $\mu P^{(i)}$ resources are used
  $\Longrightarrow$ No tasks have been reduced in $I_1$
  $\Longrightarrow$ No tasks are ready in $I_1$
- $I_2$:
- $I_3$:

# Analytical Tools

**Definitions**: The processing time [0,T] is subdivided in three sets:

- $I_1$: For all $i$, less than $\mu P^{(i)}$ resources are used
  $\Longrightarrow$ No tasks have been reduced in $I_1$
  $\Longrightarrow$ No tasks are ready in $I_1$
- $I_2$: Excludes $I_1$, for all $i$ at most $(1-\mu)P^{(i)}$ resources are used
  $\Longrightarrow$ No tasks are ready in $I_2$
  $\Longrightarrow$ At least a fraction $\mu$ of a resource is used in $I_2$
- $I_3$:

# Analytical Tools

**Definitions**: The processing time [0,T] is subdivided in three sets:

- $I_1$: For all $i$, less than $\mu P^{(i)}$ resources are used
  $\Longrightarrow$ No tasks have been reduced in $I_1$
  $\Longrightarrow$ No tasks are ready in $I_1$
- $I_2$: Excludes $I_1$, for all $i$ at most $(1-\mu)P^{(i)}$ resources are used
  $\Longrightarrow$ No tasks are ready in $I_2$
  $\Longrightarrow$ At least a fraction $\mu$ of a resource is used in $I_2$
- $I_3$: $[0, T] \setminus (I_1 \cup I_2)$
  $\Longrightarrow$ At least a fraction $1-\mu$ of a resource is used in $I_3$

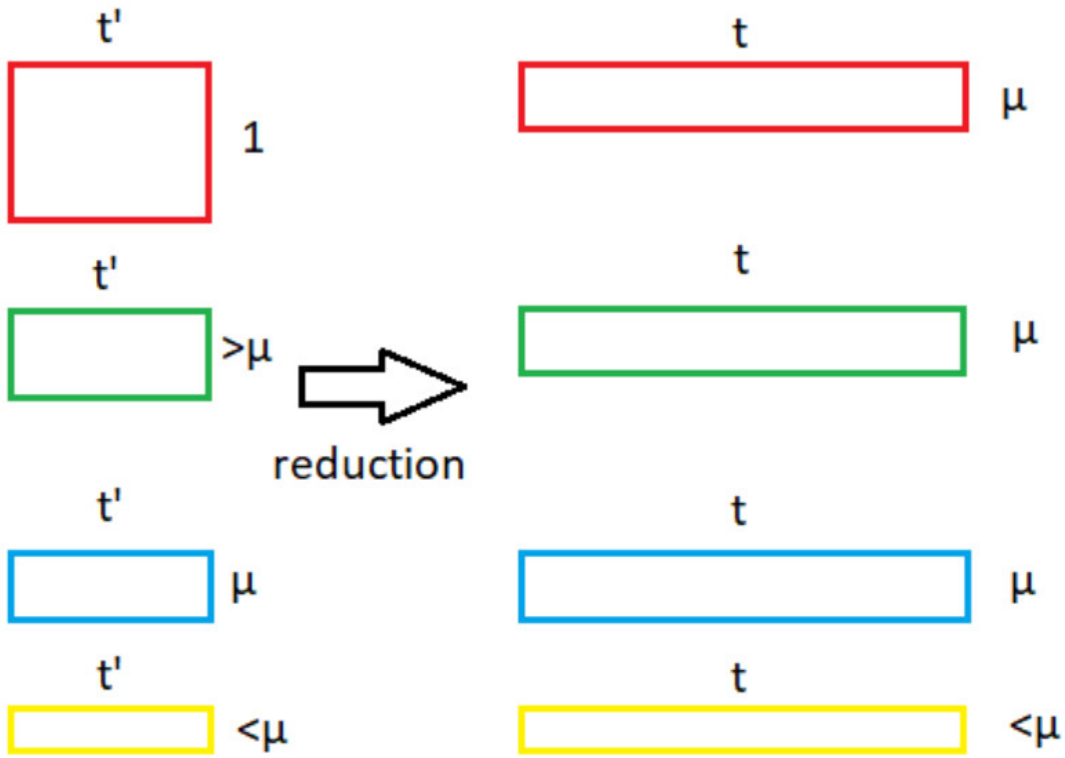
# Analytical Tools

**Definitions**: The processing time [0,T] is subdivided in three sets:

- $I_1$: For all $i$, less than $\mu P^{(i)}$ resources are used
  $\Longrightarrow$ No tasks have been reduced in $I_1$
  $\Longrightarrow$ No tasks are ready in $I_1$
- $I_2$: Excludes $I_1$, for all $i$ at most $(1-\mu)P^{(i)}$ resources are used
  $\Longrightarrow$ No tasks are ready in $I_2$
  $\Longrightarrow$ At least a fraction $\mu$ of a resource is used in $I_2$
- $I_3$: $[0, T] \setminus (I_1 \cup I_2)$
  $\Longrightarrow$ At least a fraction $1-\mu$ of a resource is used in $I_3$

$$T = T_1 + T_2 + T_3$$

# Impact of Reduction (In Resource Allocation)




After step 3:

- $t \geq t'$: The execution time may not decrease
- $t \leq \frac{t'}{\mu}$: The speedup may not be superlinear
- $\forall i, w_i \leq w'$: The work for each resource is lower than the initial work among all resources.

# Combining Two Bounds

- Critical-Path Bound:
  - There exists a path filling $I_1 \cup I_2$
  - No tasks were reduced in $I_1$

  $$\Longrightarrow T_1 + \mu T_2 \leq C(\mathbf{p}') \leq \frac{T_{opt}}{\rho} \ . \qquad (1)$$

# Combining Two Bounds

- Critical-Path Bound:
  - There exists a path filling $I_1 \cup I_2$
  - No tasks were reduced in $I_1$

  $$\Longrightarrow T_1 + \mu T_2 \leq C(\mathbf{p}') \leq \frac{T_{opt}}{\rho} \ . \qquad (1)$$

- Area Bound:
  - At least a fraction $\mu$ of a resource is used in $I_2$
  - At least a fraction $(1-\mu)$ of a resource is used in $I_3$
  - $\forall i, w_i \leq w'$

  $$\Longrightarrow \mu T_2 + (1-\mu) T_3 \leq dA(\mathbf{p}') \leq \frac{dT_{opt}}{1-\rho} \ . \qquad (2)$$

# Combining Two Bounds

- Critical-Path Bound:
  - There exists a path filling $I_1 \cup I_2$
  - No tasks were reduced in $I_1$

  $$\Longrightarrow T_1 + \mu T_2 \leq C(\mathbf{p}') \leq \frac{T_{opt}}{\rho} \ . \qquad (1)$$

- Area Bound:
  - At least a fraction $\mu$ of a resource is used in $I_2$
  - At least a fraction $(1-\mu)$ of a resource is used in $I_3$
  - $\forall i, w_i \leq w'$

  $$\Longrightarrow \mu T_2 + (1-\mu) T_3 \leq dA(\mathbf{p}') \leq \frac{dT_{opt}}{1-\rho} \ . \qquad (2)$$

**Proposition**

*Combining (1) and (2) with $T = T_1 + T_2 + T_3$, we obtain*

$$T \leq \left( \frac{1}{\rho} + \frac{d}{(1-\mu)(1-\rho)} \right) T_{\mathrm{OPT}}$$

The ratio is obtained by optimizing the two variables in above expression.

# Outline

# Conclusion

**What does this paper brings:**

- A new algorithm for multi-resource scheduling
- Several approximation ratios.

**Future work:**

- Reduce gap between upper and lower bounds of the approximation ratios
- Experimental work.