Online Scheduling of Moldable Task Graphs under Common Speedup Models (ICPP 2022)

Anne Benoit[1] Lucas Perotin (speaker)[1] Yves Robert[1,2]
Hongyang Sun[3]

[1]École Normale Supérieure de Lyon, France

[2]University of Tennessee Knoxville, USA

[3]University of Kansas, USA

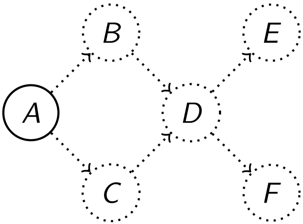
December 13, 2022

Introduction

- **Offline Scheduling vs. Online Scheduling.**
  - Offline: All tasks are known in advance,
    Goal: Find approximation ratios on polynomial algorithms.
  - Online: Tasks released on the fly,
    Goal: Derive competitive ratios against an optimal offline scheduler.
- **Independent Tasks vs. Task Graphs.**
  - Independent tasks: Tasks released and discovered on the fly,
  - Task graphs: Graph released at the start of execution,
    Tasks discovered when all predecessors are completed.

$\Rightarrow$ In this work, we focus on *online task graphs.*

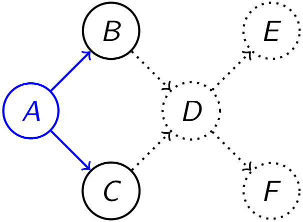
Example



- At first, only task A is known,
- Other tasks are not available yet.

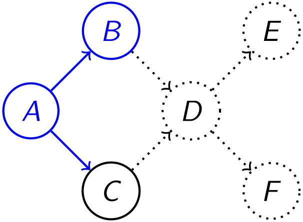
$\Longrightarrow$ The scheduler doesn't know their existence.

Example



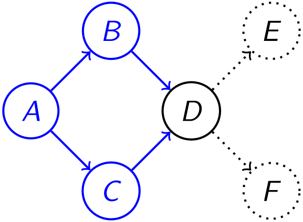
- When task A is done, the scheduler discovers tasks B and C.

Example



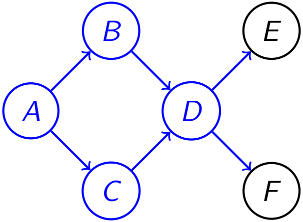
- When task B is done, task D is not discovered yet because task C is not finished.

Example



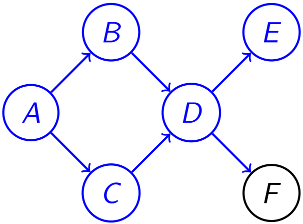
- When task C is done as well, task D becomes known and can start.

Example



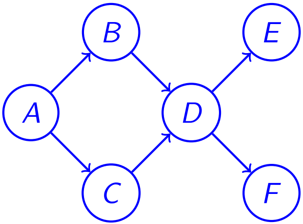
- Finally after completion of task D, tasks E and F are discovered.

Example



- Finally after completion of task D, tasks E and F are discovered and can be processed as well.

Example



- Finally after completion of task D, tasks E and F are discovered and can be processed as well.

Outline

Parallel task models

In the scheduling literature:

- **Rigid tasks**: Processor allocation is fixed.
- **Moldable tasks**: Processor allocation is decided by the system but cannot be changed.
- **Malleable tasks**: Processor allocation can be dynamically changed.

We focus on moldable tasks, because:

- They can easily adapt to the amount of available processors (contrarily to rigid tasks),
- They are easy to design/implement (contrarily to malleable tasks),
- Many computational kernels in scientific libraries are provided as moldable tasks.

Scheduling model

- Graph of n moldable tasks with precedence constraints. Each task is released when all predecessors are completed,
- P processors to process the tasks,
- Each task j's execution time $t_j(p_j)$ depends on the number of processors allocated to it and is known when the task is released,
- Area is $a_j(p_j) = p_j \times t_j(p_j)$.

Speedup models

- Roofline model: $t_j(p_j) = \frac{w_j}{\min(p_j, \bar{p}_j)}$, for some $1 \leq \bar{p}_j \leq P$.
- Communication model: $t_j(p_j) = \frac{w_j}{p_j} + (p_j - 1)c_j$,
  where c_j is the communication overhead.
- Amdahl's model: $t_j(p_j) = w_j(\frac{1-\gamma_j}{p_j} + \gamma_j)$,
  where γ_j is the inherently sequential fraction.
- General model: $t_j(p_j) = \frac{w_j(1-\gamma_j)}{\min(p_j, \bar{p}_j)} + w_j\gamma_j + (p_j - 1)c_j$,
  a combination of the three first models.
- Arbitrary model: $t_j(p_j)$ is an arbitrary function of p_j.

Scheduling objective

Scheduling objective:
Find a moldable schedule, i.e., processor allocation p_j and starting time s_j for each task j which

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

Competitive ratio:
An online algorithm is said to be *r-competitive* if its makespan T satisfies $\frac{T}{T_{\mathrm{OPT}}} \leq r$ for any task graph, where T_{OPT} denotes the best offline makespan achievable for the instance.

Main results

- New online algorithms for several common speedup models, with almost tight bounds on competitive ratios.

Model	Roofline	Comm.	Amdahl	General
Upper bound	2.62	3.61	4.74	5.72
Lower bound	2.61	3.51	4.73	5.25

- Negative result for the arbitrary speedup model. Any deterministic online algorithm is at least $\Omega(\ln(D))$-competitive where D is the length of the longest chain of the graph.

Outline

Preliminaries

Definitions: for a given processor allocation $\mathbf{p} = (p_1, p_2, \cdots, p_n)^T$

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

Preliminaries

Definitions: for a given processor allocation $\mathbf{p} = (p_1, p_2, \cdots, p_n)^T$

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

Lower bound (on makespan): $L_{min} = \max\left(\frac{A_{min}}{P}, C_{min}\right)$

Proposition

*The **optimal makespan** satisfies*

$$T_{\text{OPT}} \geq L_{\text{min}}$$

Allocation procedure

For a given μ:

- Step (1): Initial allocation
  Find an allocation $p_j \in [1, P]$ from the following optimization problem:

$$\min_{p} \ \alpha_p = \frac{a_j(p)}{a_j^{\min}}$$

$$\text{s.t. } \beta_p = \frac{t_j(p)}{t_j^{\min}} \leq \frac{1-2\mu}{\mu(1-\mu)}$$

[1] [Lepère et Al. 2OO1]

Allocation procedure

For a given μ:

- Step (1): Initial allocation
  Find an allocation $p_j \in [1, P]$ from the following optimization problem:

$$\min_p \ \alpha_p = \frac{a_j(p)}{a_j^{\text{min}}}$$

$$\text{s.t. } \beta_p = \frac{t_j(p)}{t_j^{\text{min}}} \leq \frac{1 - 2\mu}{\mu(1 - \mu)}$$

- Step (2): Adjusted allocation [1]
  If $p'_j > \lceil \mu P \rceil$ **then** $p_j \leftarrow \lceil \mu P \rceil$ **else** $p_j \leftarrow p'_j$

[1] [Lepère et Al. 2OO1]

Allocation procedure

For a given μ:

- Step (1): Initial allocation
  Find an allocation $p_j \in [1, P]$ from the following optimization problem:

$$\min_p \ \alpha_p = \frac{a_j(p)}{a_j^{\min}}$$

$$\text{s.t. } \beta_p = \frac{t_j(p)}{t_j^{\min}} \leq \frac{1-2\mu}{\mu(1-\mu)}$$

- Step (2): Adjusted allocation [1]
  If $p'_j > \lceil \mu P \rceil$ **then** $p_j \leftarrow \lceil \mu P \rceil$ **else** $p_j \leftarrow p'_j$

$\Rightarrow$ Minimize the area up to a time constraint.

$\Rightarrow$ The allocation procedure doesn't depend on the shape of the graph.

$\Rightarrow$ The best choice of μ depends on the speedup model.

[1] [Lepère et Al. 2OO1]

Scheduling algorithm and results

For a fixed processor allocation:

- List Scheduling:
  - Whenever a task is released, try to schedule it if enough processors are available
  - If not, store it in a list of available tasks.
  - Whenever an existing task completes, scan this list and schedule available tasks until no task fits (or until the list is empty).

Scheduling algorithm and results

For a fixed processor allocation:

- List Scheduling:
  - Whenever a task is released, try to schedule it if enough processors are available
  - If not, store it in a list of available tasks.
  - Whenever an existing task completes, scan this list and schedule available tasks until no task fits (or until the list is empty).
- Results: They can be summarized in the following table:

Model	Roofline	Comm.	Amdahl	General
Choice of μ	0.382	0.324	0.271	0.211
Upper bound	2.62	3.61	4.74	5.72

Outline

Analytical tools

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

- I_1: Less than μP processors 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: Less than μP processors 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 $(1-\mu)P$ processors are used.
  $\Longrightarrow$ No tasks are ready in I_2,
  $\Longrightarrow$ At least a fraction μ of a processor is used in I_2.
- I_3:

Analytical tools

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

- I_1: Less than μP processors 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 $(1 - \mu)P$ processors are used.
  $\Longrightarrow$ No tasks are ready in I_2,
  $\Longrightarrow$ At least a fraction μ of a processor is used in I_2.
- I_3: $[0, T] \setminus (I_1 \cup I_2)$.
  $\Longrightarrow$ At least a fraction $1 - \mu$ of a processor is used in I_3.

Analytical tools

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

- I_1: Less than μP processors are used.
  $\implies$ No tasks have been reduced in I_1,
  $\implies$ No tasks are ready in I_1.
- I_2: Excludes I_1, for all i at most $(1-\mu)P$ processors are used.
  $\implies$ No tasks are ready in I_2,
  $\implies$ At least a fraction μ of a processor is used in I_2.
- I_3: $[0, T] \setminus (I_1 \cup I_2)$.
  $\implies$ At least a fraction $1-\mu$ of a processor is used in I_3.

$$T = |I_1| + |I_2| + |I_3|$$

Combining two bounds

- Critical-Path Bound: *Given* $\beta = max_j(t_j(p_j)/t_j^{min})$
  - There exists a path filling $I_1 \cup I_2$,
  - No tasks were reduced in I_1, thus $t_j \leq \beta t_j^{min}$,
  - Tasks are reduced to $\lceil \mu P \rceil$ or verify $t_j \leq \beta t_j^{min}$. As $\beta \leq \frac{1}{\mu}$,

    $\Longrightarrow \frac{|I_1|}{\beta} + \mu |I_2| \leq C_{min} \leq T_{opt}$ (1)

Combining two bounds

- Critical-Path Bound: *Given* $\beta = max_j(t_j(p_j)/t_j^{min})$
  - There exists a path filling $I_1 \cup I_2$,
  - No tasks were reduced in I_1, thus $t_j \leq \beta t_j^{min}$,
  - Tasks are reduced to $\lceil \mu P \rceil$ or verify $t_j \leq \beta t_j^{min}$. As $\beta \leq \frac{1}{\mu}$,

    $\implies \frac{|I_1|}{\beta} + \mu|I_2| \leq C_{min} \leq T_{opt}$ (1)
- Area Bound: *Given* $\alpha = max_j(a_j(p_j)/a_{min})$
  - At least a fraction μ of a processor is used in I_2,
  - At least a fraction $(1-\mu)$ of a processor is used in I_3,
  - $\forall i, w_i \leq w'$.

    $\implies \mu|I_2| + (1-\mu)|I_3| \leq \frac{\alpha A_{min}}{P} \leq \alpha T_{opt}$ (2)

Combining two bounds

- **Critical-Path Bound**: *Given $\beta = max_j(t_j(p_j)/t_j^{min})$*
  - There exists a path filling $I_1 \cup I_2$,
  - No tasks were reduced in I_1, thus $t_j \leq \beta t_j^{min}$,
  - Tasks are reduced to $\lceil \mu P \rceil$ or verify $t_j \leq \beta t_j^{min}$. As $\beta \leq \frac{1}{\mu}$,

    $\Longrightarrow \frac{|I_1|}{\beta} + \mu|I_2| \leq C_{min} \leq T_{opt}$ (1)

- **Area Bound**: *Given $\alpha = max_j(a_j(p_j)/a_{min})$*
  - At least a fraction μ of a processor is used in I_2,
  - At least a fraction $(1-\mu)$ of a processor is used in I_3,
  - $\forall i, w_i \leq w'$.

    $\Longrightarrow \mu|I_2| + (1-\mu)|I_3| \leq \frac{\alpha A_{min}}{P} \leq \alpha T_{opt}$ (2)

Proposition

Combining (1) and (2) with $T = |I_1| + |I_2| + |I_3|$, and given $\beta \leq \frac{1-2\mu}{\mu(1-\mu)}$, we have:

$$T \leq \frac{\mu\alpha + 1 - 2\mu}{\mu(1-\mu)} \times T_{\mathrm{OPT}}.$$

Obtaining final results

Proposition

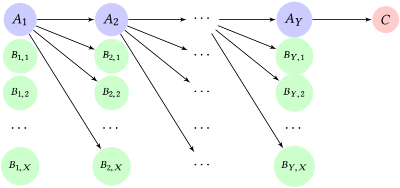
Combining (1) and (2) with $T = |I_1| + |I_2| + |I_3|$, and given $\beta \leq \frac{1-2\mu}{\mu(1-\mu)}$, we have:

$$T \leq \frac{\mu\alpha + 1 - 2\mu}{\mu(1-\mu)} \times T_{\text{OPT}}.$$

- For each speedup model, find a good upperbound on α as a function of μ.
- Find the μ minimizing the ratio.

Outline

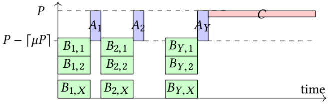
Lower bound for common speedup models



Tasks parameters are chosen so that:

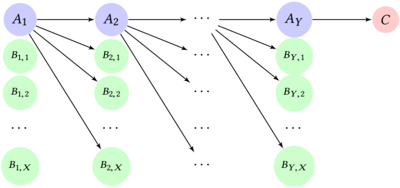
- It is barely impossible to process a full layer in parallel;
- Area of the tasks in B is as high as possible, other areas are negligibles;
- C is as long as possible.

$\Rightarrow$ Processor utilization is as bad as possible
$\Rightarrow$ Area and Critical Path are as bad as possibles.
A possible result of our heurisitic is shown in (a)



(a)

Lower bound for common speedup models

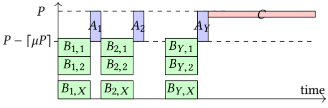


Tasks parameters are chosen so that:

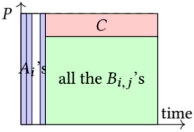
- It is barely impossible to process a full layer in parallel;
- Area of the tasks in B is as high as possible, other areas are negligibles;
- C is as long as possible.

⇒ Processor utilization is as bad as possible
⇒ Area and Critical Path are as bad as possibles.

A possible result of our heurisitic is shown in (a), and the optimal is shown in (b).

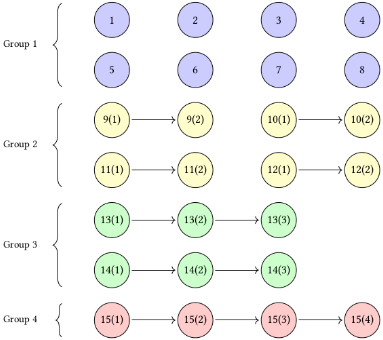


(a)



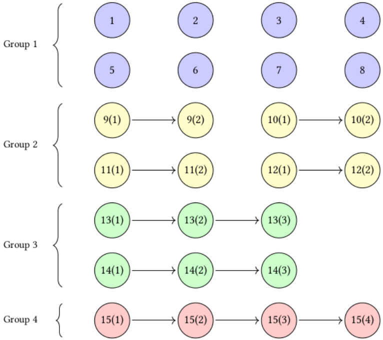
(b)

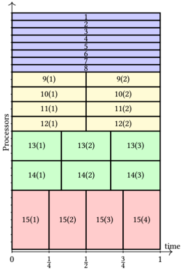
Instance and trick for negative result



- $K = 2^l$ groups of indistinguishable tasks,
- With 2^{i-1} processors for tasks in group i, the processing time is 1,

Instance and trick for negative result

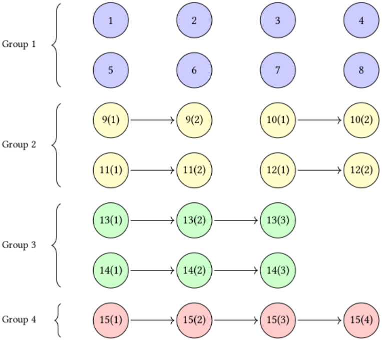




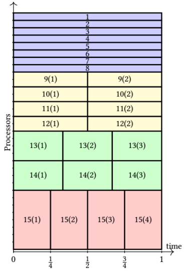
(a)

- $K = 2^l$ groups of indistinguishable tasks,
- With 2^{i-1} processors for tasks in group i, the processing time is 1,

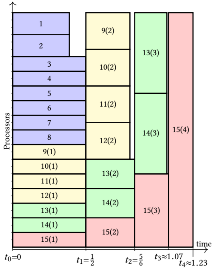
Instance and trick for negative result



- $K = 2^l$ groups of indistinguishable tasks,
- With 2^{i-1} processors for tasks in group i, the processing time is 1,
- One rep. of all task of group i is longer than $\frac{1}{l+i}$,
- Any deterministic algorithm could produce a schedule of length at least $\Omega(\ln(K))$.



(a)



(b)

Outline

Conclusion

What does this paper bring:

- A new algorithm for online scheduling of moldable task graphs,
- Almost tight competitive ratios for several common speedup models.

Current work:

- Improve the allocation analysis,
- Close gap between upper and lower bounds for competitive ratio,
- Build a lower bound for algorithms with two-phases approach (allocation independent from schedule)
- Experimental evaluation.