Scheduling on Hybrid Platforms: Improved Approximability Window

Vincent Fagnon[1] Imed Kacem[2]
Giorgio Lucarelli[2] **Bertrand Simon**[3]

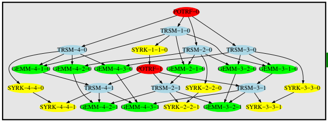
1: LIG, Univ. Grenoble Alpes (France).
2: LCOMS, Univ. Lorraine (France).
3: IN2P3 Computing Center (France).

LATIN – January 2021

```
FOR k = 0..TILES-1
  FOR n = 0..k-1
    A[k][k]<-DSYRK(A[k][n],A[k][k])
  A[k][k]<-DPOTRF(A[k][k])
  FOR m = k+1..TILES-1
    FOR n = 0..k-1
      A[m][k]<-DGEMM(A[k][n],A[m][n],A[m][k])
    A[m][k]<-DTRSM(A[k][k],A[m][k])
```

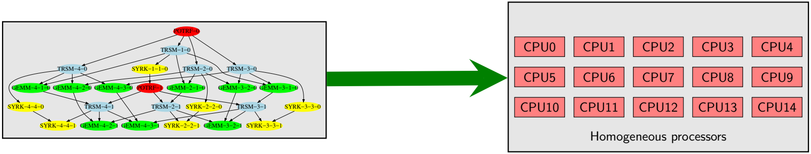
Runtime Software

- Schedule tasks
- Aware of durations
- Objective: complete the graph ASAP (minimize the makespan)



Runtime Software

- Schedule tasks
- Aware of durations
- Objective: complete the graph ASAP (minimize the makespan)



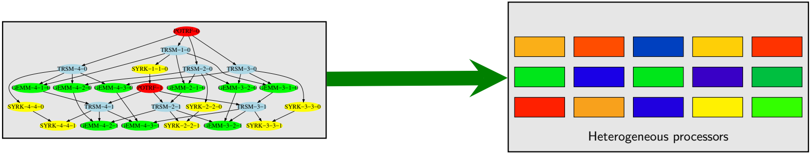
Runtime Software

- Schedule tasks
- Aware of durations
- Objective: complete the graph ASAP (minimize the makespan)

Classical models

- m identical processors: $P|prec|C_{max}$ - ignore accelerators (GPUs...)

Scheduling on computing platforms



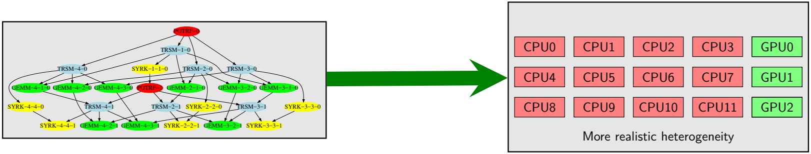
Runtime Software

- Schedule tasks
- Aware of durations
- Objective: complete the graph ASAP (minimize the makespan)

Classical models

- m identical processors: $P|prec|C_{max}$ - ignore accelerators (GPUs...)
- m unrelated processors: $R \mid prec \mid C_{max}$ - too complex

Scheduling on computing platforms



Runtime Software

- Schedule tasks
- Aware of durations
- Objective: complete the graph ASAP (minimize the makespan)

Classical models

- m identical processors: $P|prec|C_{max}$ - ignore accelerators (GPUs...)
- m unrelated processors: $R \mid prec \mid C_{max}$ - too complex

Our model

- Two types of processors: e.g., m CPUs and k GPUs

Formal definition

Input

- Graph of tasks with precedence constraints
- m identical processors (CPU) and k identical processors (GPU)
- $m \geq k$
- For each task: its running time $\overline{p_i}$ on CPU and $\underline{p_i}$ on GPU

Output

- Schedule of minimum makespan (ignoring communication times)

Evaluation metric

- Approximation factor: maximum value of $\dfrac{\text{obtained makespan}}{\text{optimal makespan}}$

Note: other scenarios studied (independent tasks, online arrival of tasks)

Some intuition

Two sub-problems

- allocation: on which processor type (CPU/GPU) we place each task
- schedule: once the allocation is fixed, when each task is run

Assume the allocation is fixed

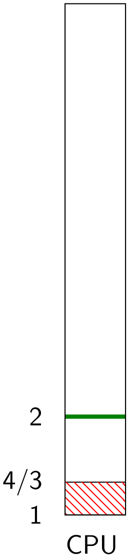
Best known algorithm: list-scheduling [Graham '66]
(if a processor is available and a task can run on it, do it)
We can define:

- W_C (resp. W_G): total load on CPUs (resp. GPUs)
- CP: time to complete the longest path (critical path)

Lemma

List-scheduling makespan is at most $CP + \frac{W_C}{m} + \frac{W_G}{k}$ *so it is a 3-approximation when the allocation is fixed.*

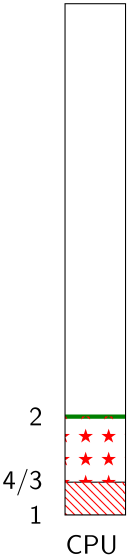
Approximation window – a) identical processors



List-Scheduling [Graham '66]

NP-hard [Lenstra & Rinnooy Kan '78]

Approximation window – a) identical processors

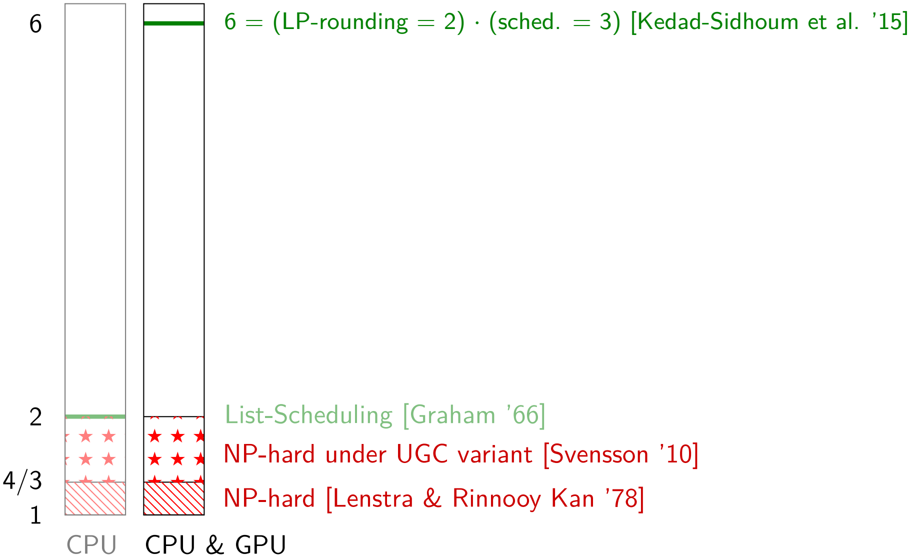


List-Scheduling [Graham '66]

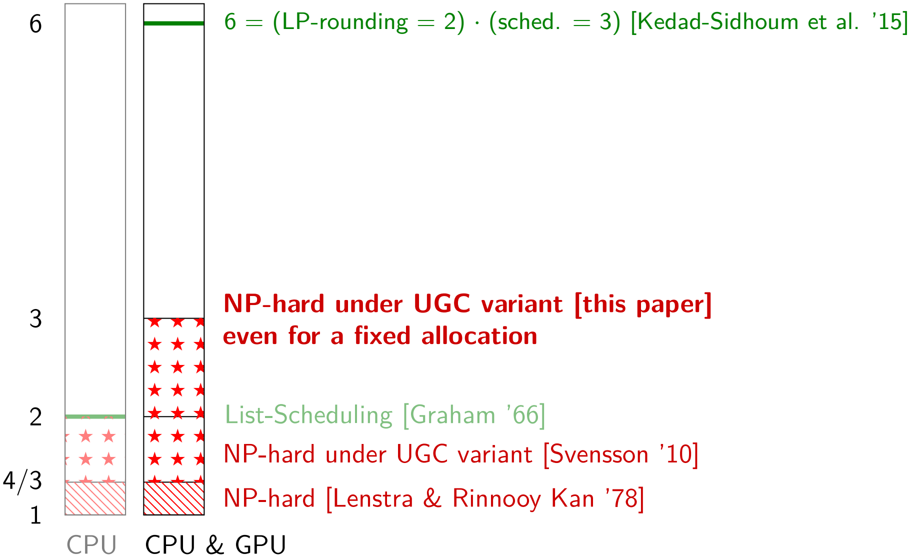
NP-hard under UGC variant [Svensson '10]

NP-hard [Lenstra & Rinnooy Kan '78]

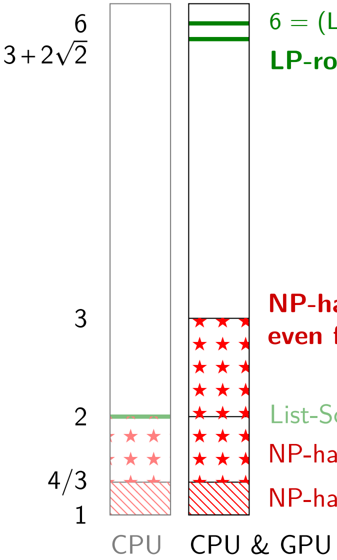
Approximation window – b) CPUs & GPUs



Approximation window – b) CPUs & GPUs



Approximation window – b) CPUs & GPUs



$6 = (\text{LP-rounding} = 2) \cdot (\text{sched.} = 3)$ [Kedad-Sidhoum et al. '15]

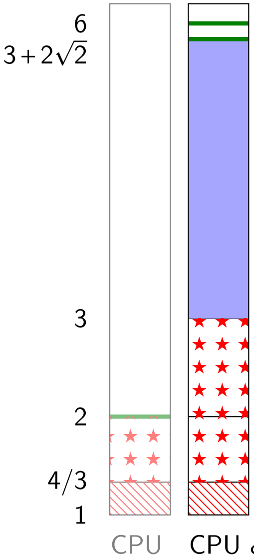
LP-rounding + subtile analysis [this talk]

NP-hard under UGC variant [this paper] even for a fixed allocation

List-Scheduling [Graham '66]

NP-hard under UGC variant [Svensson '10]

NP-hard [Lenstra & Rinnooy Kan '78]



$6 = (\text{LP-rounding} = 2) \cdot (\text{sched.} = 3)$ [Kedad-Sidhoum et al. '15]

LP-rounding + subtile analysis [this talk]

Open Question:
how much do we lose to decide the allocation?

NP-hard under UGC variant [this paper]
even for a fixed allocation

List-Scheduling [Graham '66]

NP-hard under UGC variant [Svensson '10]

NP-hard [Lenstra & Rinnooy Kan '78]

A 6-approximation algorithm [Kedad-Sidhoum et al, '15]

Idea: find an allocation by rounding an LP solution,
then use List Scheduling.

minimize: C

$$\frac{1}{m}\mathbf{W_C} \leq C$$

$$\frac{1}{k}\mathbf{W_G} \leq C$$

$$\mathbf{CP} \leq C$$

$=$

minimize: C

$$\frac{1}{m}\sum_j \overline{p_j} x_j \leq C$$

$$\frac{1}{k}\sum_j \underline{p_j}(1 - x_j) \leq C$$

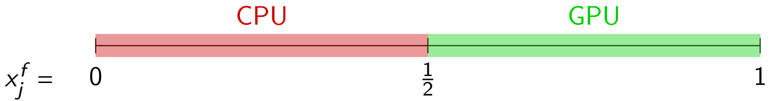
$$C_i + \overline{p_j} x_j + \underline{p_j}(1 - x_j) \leq C_j \text{ for all i} \rightarrow \text{j}$$

$$0 \leq C_j \leq C$$

$$x_j \in [0, 1]$$

x_j: equals 0 (resp. 1) if task j goes to CPU (resp. GPU)

Rounding the LP [Kedad-Sidhoum et al, '15]



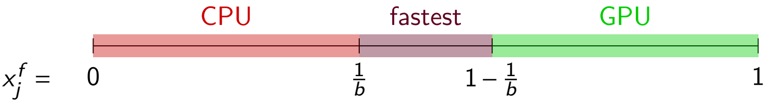
Hence, we have ($\cdot^f \rightarrow$ optimal fractional solution):

$$
\begin{aligned}
C_{max} &\leq \frac{1}{m}\mathbf{W_C} + \frac{1}{k}\mathbf{W_G} + \mathbf{CP} \\
&\leq 2 \cdot \frac{1}{m} W_C^f + 2 \cdot \frac{1}{k} W_G^f + 2 \cdot CP^f \\
&\leq 6 \cdot OPT
\end{aligned}
$$

Tight in two ways:

- approximation factor $= 6$ (reached on an example)
- LP integrality gap $= 2$

Better rounding of the LP ($b > 2$)



We gain on the critical path: $\mathbf{CP} \leq \frac{b}{b-1} CP^f$,

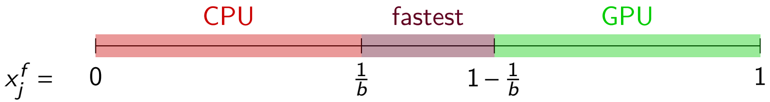
and lose on the loads: $\frac{1}{m}\mathbf{W_C} + \frac{1}{k}\mathbf{W_G} \leq b \cdot \left(\frac{1}{m} W_C^f + \frac{1}{k} W_G^f\right)$.

Hence,

$$C_{max} \leq \frac{1}{m}\mathbf{W_C} + \frac{1}{k}\mathbf{W_G} + \mathbf{CP}$$

$$\leq$$

Better rounding of the LP ($b > 2$)



We gain on the critical path: $\mathbf{CP} \leq \frac{b}{b-1} CP^f$,

and lose on the loads: $\frac{1}{m}\mathbf{W_C} + \frac{1}{k}\mathbf{W_G} \leq b \cdot \left(\frac{1}{m} W_C^f + \frac{1}{k} W_G^f\right)$.

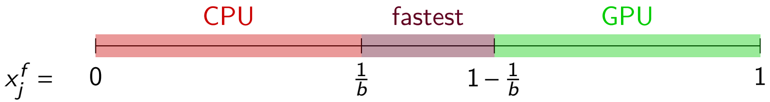
But we have $\mathbf{W_C} + \mathbf{W_G} \leq \frac{b}{b-1}\left(W_C^f + W_G^f\right)$

Hence,

$$C_{max} \leq \frac{1}{m}\mathbf{W_C} + \frac{1}{k}\mathbf{W_G} + \mathbf{CP}$$

$$\leq$$

Better rounding of the LP ($b > 2$)



We gain on the critical path: $\mathbf{CP} \leq \frac{b}{b-1} CP^f$,

and lose on the loads: $\frac{1}{m}\mathbf{W_C} + \frac{1}{k}\mathbf{W_G} \leq b \cdot \left(\frac{1}{m} W_C^f + \frac{1}{k} W_G^f\right)$.

But we have $\mathbf{W_C} + \mathbf{W_G} \leq \frac{b}{b-1}\left(W_C^f + W_G^f\right)$

Hence,

$$\begin{aligned} C_{max} &\leq \frac{1}{m}\mathbf{W_C} + \frac{1}{k}\mathbf{W_G} + \mathbf{CP} \\ &\leq \frac{1}{m}(\mathbf{W_C} + \mathbf{W_G}) + \frac{m-k}{mk}\mathbf{W_G} + \mathbf{CP} \\ &\leq \cdots \leq (3 + 2\sqrt{2}) \cdot OPT \ \text{ for } b = 1 + \sqrt{2} \end{aligned}$$

Few words on the conditional lower bound

Same assumption as in [Bazzi, Norouzi-Fard '15], a variant of the UGC stronger than the one used in [Svensson '10, Bansal & Khot '09]

- Stays valid when the allocation is fixed
- Stays valid if processors are *related*: $\frac{\overline{p_i}}{\underline{p_i}}$ is the same for all tasks
- Stays valid for unrelated processors and any value of m/k

Note: m/k linked to the online problem difficulty (best $= \Theta\left(\sqrt{\frac{m}{k}}\right)$)
[Amaris, Lucarelli, Mommessin, Trystram '19
Canon, Marchal, Simon, Vivien '19]

Approximation ratio in function of m/k