TD Optimisation – session 4 – Scheduling Independent Tasks

17 octobre 2025

Learning objective : find extreme instances showing the tightness of approximation ratios for classical scheduling strategies and their behaviors for the problem $P||C_{\max}$.

> The first 4 exercises are essential.

Exercice 1 : $(2 - 1/m)$ ratio for LST

Recall that LST has two properties :

1. task order is arbitrary ;
2. each task is put on the first available processor.

To identify a worst-case instance, we need to define :
— the number of processors m ;
— the number of tasks n, their costs and their order.
Recall the worst-case instance with $m = 2$. What is the worst-case instance with $m = 3$?

Hint : start by considering an arbitrary instance with a few tasks, draw the Gantt diagram obtained with LST and compare it with the optimal one.

> Pour $m = 2$:
> — $\{p_i\} = \{1, 1, 2\}$
> — $C_{\max}(LST)/C_{\max}(OPT) = 3/2$
> Pour $m = 3$:
> — $\{p_i\} = \{1, 1, 1, 1, 1, 1, 3\}$
> — $C_{\max}(LST)/C_{\max}(OPT) = 5/3$

Exercice 2 : $(4/3 - 1/3m)$ ratio for LPT

Recall that LPT has two properties :

1. task are ordered by decreasing cost ;
2. each task is put on the first available processor.

To identify a worst-case instance, we need to define :
— the number of processors m ;
— the number of tasks n and their costs.
What is the worst-case instance with $m = 2$?

Hint : the objective is to build an instance for which the optimal schedule is different than the one given by LPT. While the largest task is necessarily on any processor in both cases, the second largest task can be on the same processor in the optimal solution while it is necessarily on the other processor for LPT.

> As a second hint (before looking the solution below), the worst-case instances has 2 long tasks (on the same processor in the optimal schedule) and 3 shorter tasks (on the other processor).
> — $m = 2$
> — $\{p_i\} = \{3, 3, 2, 2, 2\}$
> — $C_{\max}(LPT)/C_{\max}(OPT) = 7/6$

Exercice 3 : 13/11 ratio for MULTIFIT

MULTIFIT relies on a binary search to find the best possible objective value (makespan). For each considered makespan, it uses FFD to try producing a valid schedule. FFD considered tasks by decreasing cost and assign each of them to the first processor (starting with only one processor and adding a new one anytime a task does not fit). If it ends up with more than m processors, the schedule is invalid.

To identify a worst-case instance, we need to define :
— the number of processors m ;
— the number of tasks n and their costs.
Find an instance for which MULTIFIT is not optimal ?
Hint : with $m = 2$ processors, the worst-case ratio is $\frac{8}{7}$.

As a second hint (before looking the solution below), the worst-case instances has 2 long tasks (on different processors in the optimal schedule) and 4 shorter tasks. This is actually the opposite of the worst-case instance for LPT.

— $m = 2$
— $\{p_i\} = \{3, 3, 2, 2, 2, 2\}$
— when trying the optimal makespan 7, FFD schedules both longest tasks on the same processors and fails to schedule the last one
— $C_{\max}(MULTIFIT)/C_{\max}(OPT) = 8/7$

Exercice 4 : Proof of 2-approximation ratio for MULTIFIT

By observing the basic bounds to achieve the 2-approximation ratio for the list heuristics, provide the reasoning to prove that MULTIFIT heuristic is also a 2-approximation algorithm. We can first prove a result on FFD and deduce the 2-approximation ratio from it. What is the FFD property we must prove to show that MULTIFIT is a 2-approximation ? How can we prove this property (recall and use the 2 basics bounds on $C_{\max}(OPT)$ for this problem) ?

To show that MULTIFIT is a 2-approximation, we must show that FFD always succeeds to schedule all the tasks with a candidate makespan of $2C_{\max}(OPT)$.
The basic bounds we need are $\max_j p_j \leq C_{\max}(OPT)$ and $\sum_j p_j \leq mC_{\max}(OPT)$.
We can now proceed by contradiction : assume that FFD fails to schedule at least one task when the candidate makespan is $2C_{\max}(OPT)$.
In this case, there is not enough available time on all processors when this task is considered. We can deduce that all processors are busy until at least $C_{\max}(OPT)$ by observing that even with the longest task (shorter tasks would lead to the same conclusion), $\max_j p_j \leq C_{\max}(OPT)$.
If all processors are busy until at least $C_{\max}(OPT)$, then the sum of the time spent processing tasks is greater than $mC_{\max}(OPT)$, which is greater than or equal to the sum of all processing times (second basic bounds). This means that all tasks have been processed, leading to a contradiction because we assumed that FFD failed to schedule one task.

Exercice 5 : SLACK

SLACK est un algorithme plus récent (Della Croce et Scatamacchia, 2020) basé sur la stratégie gloutonne suivante :
— Trier les tâches par ordre décroissant de leur taille ;
— Découper l'ensemble trié en tuples de m tâches ;
— Soit "slack" la différence entre la taille du premier job et la taille du dernier job de chaque tuple ;
— Trier l'ensemble des tuples dans l'ordre décroissant de leur "slack".
SLACK a une complexité en temps de $O(n \log(n) + n \log(m))$.

Algorithme 1 : SLACK

Data : instance de $P||C_{\max}$, avec
m machines,
n jobs et leur temps d'exécution

1 réindexer les jobs, de manière à obtenir $p_1 \geq p_2 \geq ... \geq p_n$

2 découper l'ensemble obtenu en $\lceil \frac{n}{m} \rceil$ tuples de m jobs (ajout de jobs "dummy" de taille nulle pour le dernier tuple, si n n'est pas un multiple de m)

3 considérer chaque tuple avec la différence de temps ("Slack") entre le premier job du tuple et le dernier.

4

$$\left\{ \begin{array}{lll} \{1,\ldots,m\} & \{m+1,\ldots,2\cdot m\} & \ldots \\ p_1 - p_m & p_{m+1} - p_{2\cdot m} & \ldots \end{array} \right\}$$

trier les tuples par ordre décroissant de "Slack" et ainsi former un nouvel ensemble `// e.g:` $\{\{m+1,...,2\cdot m\}\{1,...,m\}\}$ `si` $p_{m+1} - p_{2\cdot m} > p_1 - p_m$.

5 appliquer l'ordonnancement (affectation à la machine la moins chargée à ce moment là) à l'ensemble ainsi obtenu.

Dérouler SLACK sur un exemple simple avec $m = 3$ et $\{p_i\} = \{11, 10, 7, 6, 5, 5, 5, 3, 2\}$.

Trouver une instance donnant un makespan différent avec SLACK et LPT.

Indice : LPT se comporte mal si les plus petites tâches (les dernières considérées) sont très hétérogènes.

- $m = 2$
- $n = 6$
- $\{p_i\} = \{5, 4, 4, 3, 3, 1\}$
- $C_{\max}(SLACK) = 9$
- $C_{\max}(LPT) = 11$

Exercice 6 : Proof of 2-approximation ratio for LST

Provide the basic bounds to achieve the 2-approximation ratio for list heuristics. Which properties of those heuristics is noteworthy for this proof. Provide the reasoning of the rest of the proof.

See lecture.