M1 Info - Optimisation et Recherche Opérationnelle

Cours 5 - Algorithmes d'approximation

Equilibrage de charges

Semestre Automne 2021-2022 - Université Claude Bernard Lyon 1

Christophe Crespelle
christophe.crespelle@univ-lyon1.fr

département
Informatique

Faculté des Sciences et Technologies
Université Claude Bernard Lyon 1

Problème de l'equilibrage de charges

Entrée : m machines, n taches avec chacune un temps d'execution de $t_i \in \mathbb{R}_+^*, 1 \leq i \leq n$.

Contraintes :

- taches insecables : effectuees entierement sur une machine
- une machine ne peut faire qu'une tache a la fois

Sortie : une affectation des taches aux m machines telle que la machine qui termine en dernier termine le plus tot possible

Exemple : m=3, n=5, $t_1 = 2$, $t_2 = 3$, $t_3 = 4$, $t_4 = 3$, $t_5 = 5$

Problème de l'equilibrage de charges

Entrée : m machines, n taches avec chacune un temps d'execution de $t_i \in \mathbb{R}_+^*, 1 \leq i \leq n$.

Contraintes :

- taches insecables : effectuees entierement sur une machine
- une machine ne peut faire qu'une tache a la fois

Sortie : une affectation des taches aux m machines telle que la machine qui termine en dernier termine le plus tot possible

Difficulte de calcul : **NP-complet**

NP-difficile

Problème de l'equilibrage de charges

Entrée : m machines, n taches avec chacune un temps d'execution de $t_i \in \mathbb{R}_+^*, 1 \leq i \leq n$.

Contraintes :

- taches insecables : effectuees entierement sur une machine
- une machine ne peut faire qu'une tache a la fois

Sortie : une affectation des taches aux m machines telle que la machine qui termine en dernier termine le plus tot possible

Difficulte de calcul : **NP-complet**

On va faire un algorithme polynomial qui donne une solution **approchee garantie**.

Algorithme d'approximation

On note I l'instance du probleme, $OPT(I)$ la valeur de la solution optimale sur I et $ALG(I)$ la valeur retournee par l'algorithme sur I.

Définition

Pour un probleme de minimisation, un algorithme d'approximation avec ratio d'approximation $\rho \geq 1$ est tel que $\forall I$, $\frac{ALG(I)}{OPT(I)} \leq \rho$.

Pour un probleme de maximisation, un algorithme d'approximation avec ratio d'approximation $\rho \geq 1$ est tel que $\forall I$, $\frac{OPT(I)}{ALG(I)} \leq \rho$.

min $ALG(I) \leq \rho \, OPT(I)$

max $OPT(I) \leq \rho \, ALG(I)$

Algorithme d'approximation

On note I l'instance du probleme, $OPT(I)$ la valeur de la solution optimale sur I et $ALG(I)$ la valeur retournee par l'algorithme sur I.

Définition

Pour un probleme de minimisation, un algorithme d'approximation avec ratio d'approximation $\rho \geq 1$ est tel que $\forall I, \frac{ALG(I)}{OPT(I)} \leq \rho$.

Pour un probleme de maximisation, un algorithme d'approximation avec ratio d'approximation $\rho \geq 1$ est tel que $\forall I, \frac{OPT(I)}{ALG(I)} \leq \rho$.

Exemple : Un probleme de minimisation et un algorithme pour le resoudre. Sur trois instances I, I', I'', on observe :

- $OPT(I) = 3$ et $ALG(I) = 6$
- $OPT(I') = 8$ et $ALG(I') = 12$
- $OPT(I'') = 6$ et $ALG(I'') = 8$

Quel est le ratio d'approximation de cet algorithme ?

Algorithme d'approximation

On note I l'instance du probleme, $OPT(I)$ la valeur de la solution optimale sur I et $ALG(I)$ la valeur retournee par l'algorithme sur I.

Définition

Pour un probleme de minimisation, un algorithme d'approximation avec ratio d'approximation $\rho \geq 1$ est tel que $\forall I, \frac{ALG(I)}{OPT(I)} \leq \rho$.

Pour un probleme de maximisation, un algorithme d'approximation avec ratio d'approximation $\rho \geq 1$ est tel que $\forall I, \frac{OPT(I)}{ALG(I)} \leq \rho$.

Exemple : pour l'equilibrage de charges on va faire un algo polynomial tel que la solution retournee par l'algorithme verifie toujours $ALG(I) \leq 2OPT(I)$: l'algorithme a un ratio d'approximation 2.

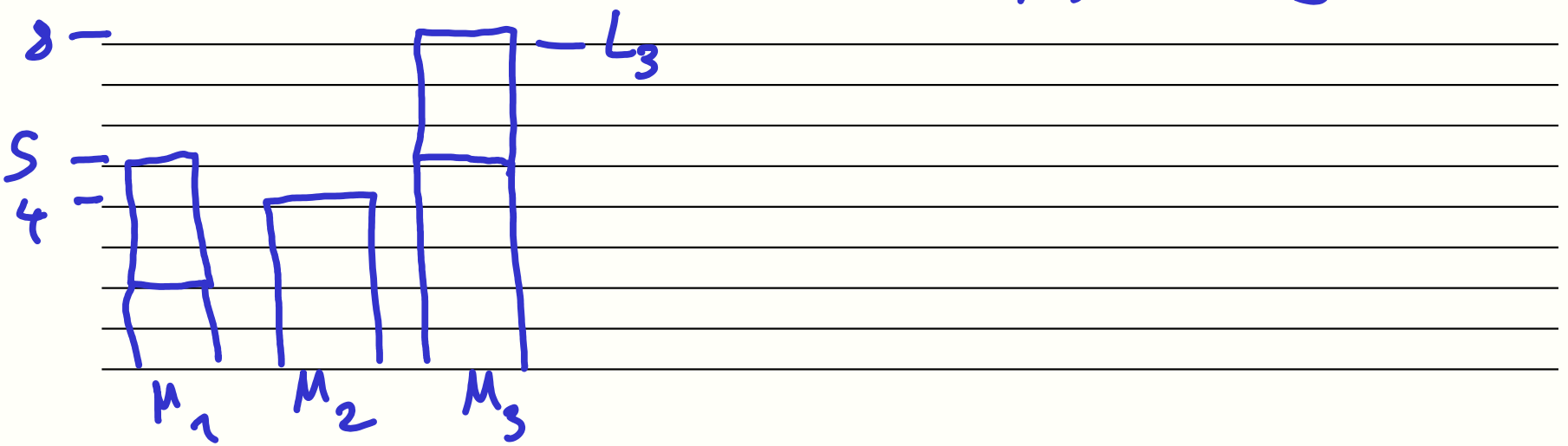
Equilibrage de charges

Définition

Pour $1 \leq i \leq m$, on note $J(i)$ l'ensemble des taches affectees a la machine i.

La charge de la machine i est definie comme $L_i = \sum_{j \in J(i)} t_j$.

$J(1) = \{1, 2\}$ $J(2) = \{3\}$ $J(3) = \{4, 5\}$ $L_1 = 5$ $L_2 = 4$ $L_3 = 9$



Définition

Le **makespan** *d'une affectation est la charge maximum d'une machine dans cette affectation : objectif a minimiser.*

Le makespan minimum sur toutes les affectations sera note L^*.

Premier algorithme - List Scheduling

ALGO :

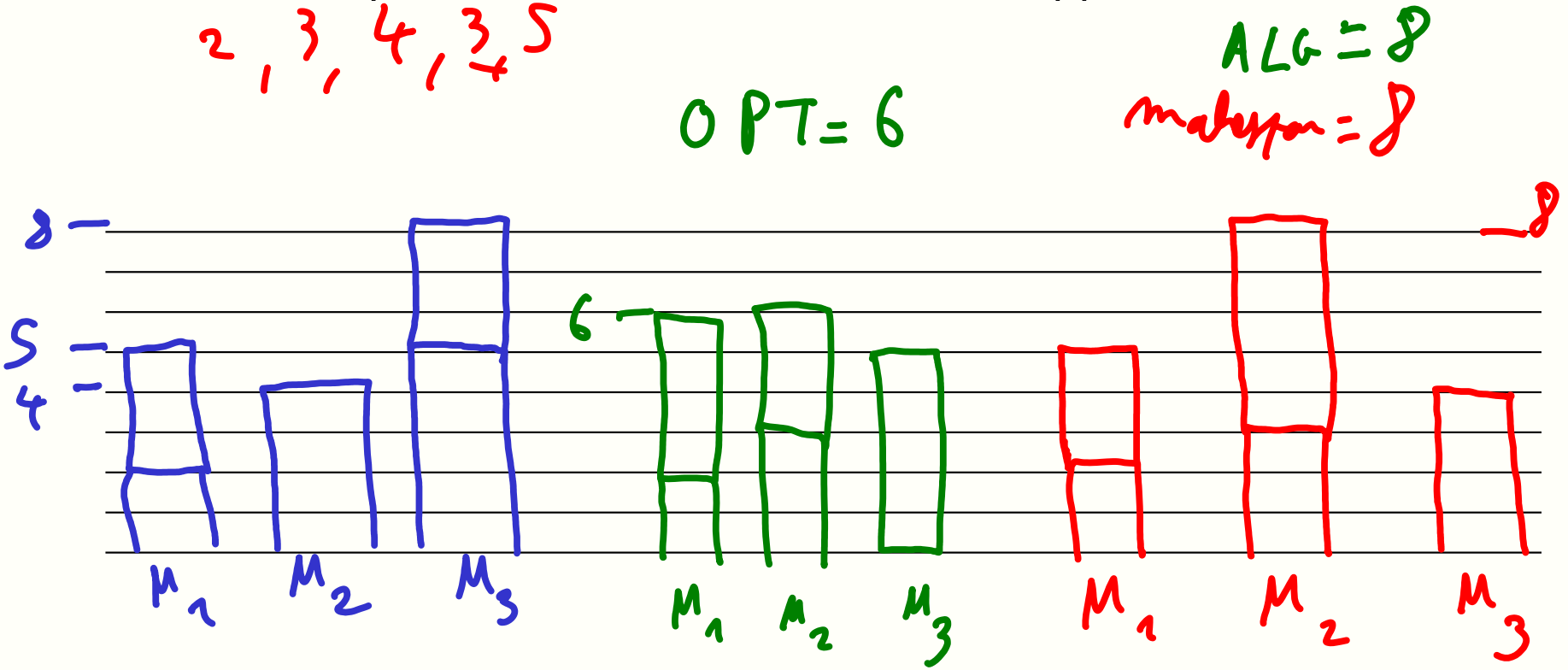
- considerer les taches une par une dans un **ordre quelconque**
- pour chaque tache l'affecter a la machine la moins chargee

Premier algorithme - List Scheduling

ALGO :

- considerer les taches une par une dans un **ordre quelconque**
- pour chaque tache l'affecter a la machine la moins chargee

Ca c'est simple ! ... et ca donne un ratio d'approximation 2 ! ! !



Premier algorithme - List Scheduling

ALGO :

- considerer les taches une par une dans un **ordre quelconque**
- pour chaque tache l'affecter a la machine la moins chargee

Ca c'est simple ! ... et ca donne un ratio d'approximation 2 ! ! !

Le pompon : en online ca marche aussi !

Premier algorithme - List Scheduling

Conseil general pour prouver les ratio d'algo d'approx :

Montrer que la solution optimale ne peut pas etre trop bonne, pas meilleure que...

Donc dans ce cas il faut minorer L^*.

Premier algorithme - List Scheduling

Conseil general pour prouver les ratio d'algo d'approx :

Montrer que la solution optimale ne peut pas etre trop bonne, pas meilleure que...
Donc dans ce cas il faut minorer L^*.

Lemme

Le makespan minimum $L^* \geq \max_{i=1}^{n}\{t_i\}$.

Premier algorithme - List Scheduling

Conseil general pour prouver les ratio d'algo d'approx :

Montrer que la solution optimale ne peut pas etre trop bonne, pas meilleure que...

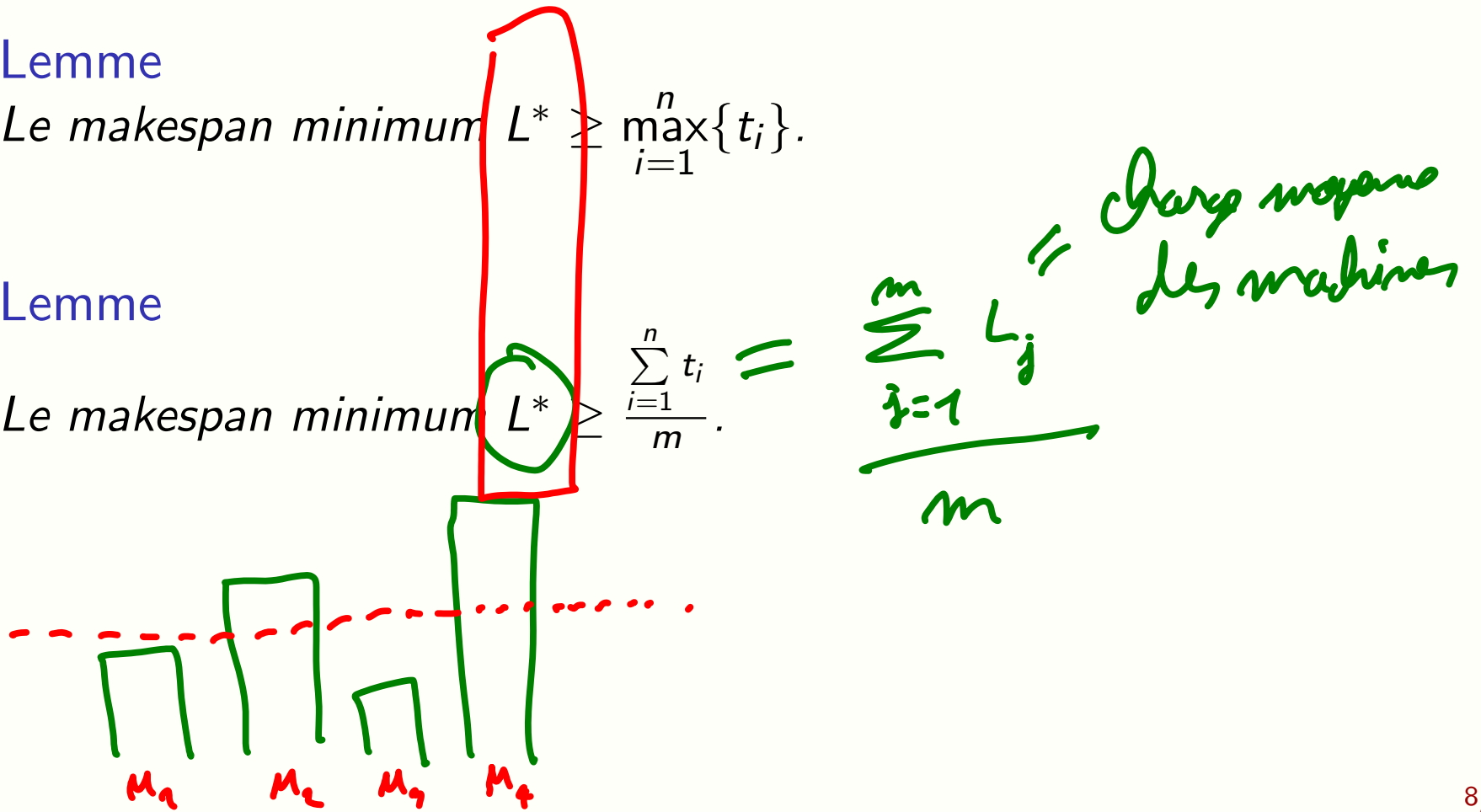
Donc dans ce cas il faut minorer L^*.

Lemme

Le makespan minimum $L^* \geq \max_{i=1}^{n}\{t_i\}$.

Lemme

Le makespan minimum $L^* \geq \frac{\sum_{i=1}^{n} t_i}{m}$.



Preuve du ratio d'approx

Théorème

L'algo List Scheduling a un ratio d'approximation 2.

Preuve du ratio d'approx

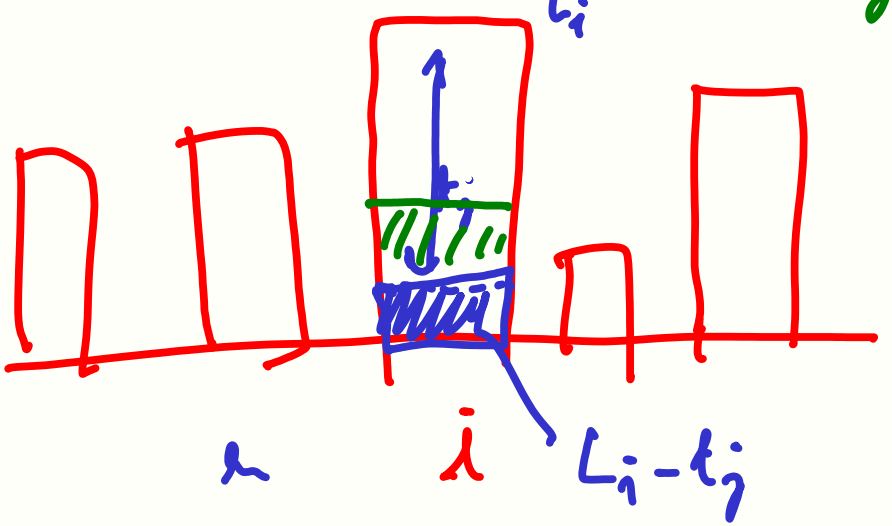
Théorème

L'algo List Scheduling a un ratio d'approximation 2.

Démonstration.

Soit i machine avec la plus grande charge et soit j la dernière tache qui lui a ete affectee.

On a $\forall k \in [\![1, m]\!], L_i - t_j \leq L_k$.



□

Preuve du ratio d'approx

Théorème

L'algo List Scheduling a un ratio d'approximation 2.

Démonstration.

Soit i machine avec la plus grande charge et soit j la derniere tache qui lui a ete affectee.

On a $\forall k \in [\![1, m]\!], L_i - t_j \leq L_k$.

D'ou, $(L_i - t_j) \leq \frac{1}{m} \sum_{i=1}^{m} L_i = \frac{1}{m} \sum_{i=1}^{n} t_i$.

□

Preuve du ratio d'approx

Théorème

L'algo List Scheduling a un ratio d'approximation 2.

Démonstration.

Soit i machine avec la plus grande charge et soit j la derniere tache qui lui a ete affectee.

On a $\forall k \in [\![1, m]\!], L_i - t_j \leq L_k$.

D'ou, $(L_i - t_j) \leq \frac{1}{m} \sum_{i=1}^{m} L_i = \frac{1}{m} \sum_{i=1}^{n} t_i$.

C.a.d. $L_i - t_j \leq \frac{1}{m} \sum_{i=1}^{n} t_i \leq L^*$ d'apres le lemme.

□

Preuve du ratio d'approx

Théorème

L'algo List Scheduling a un ratio d'approximation 2.

Démonstration.

Soit i machine avec la plus grande charge et soit j la derniere tache qui lui a ete affectee.

On a $\forall k \in [\![1, m]\!], L_i - t_j \leq L_k$.

D'ou, $(L_i - t_j) \leq \frac{1}{m} \sum_{i=1}^{m} L_i = \frac{1}{m} \sum_{i=1}^{n} t_i$.

C.a.d. $L_i - t_j \leq \frac{1}{m} \sum_{i=1}^{n} t_i \leq L^*$ d'apres le lemme.

Donc, $L_i \leq L^* + t_j \leq 2L^*$, d'apres l'autre lemme. □

A-t-on prouve le meilleur ratio ?

- on a prouve $\rho \leq 2$, mais a-t-on $\rho < 2$?

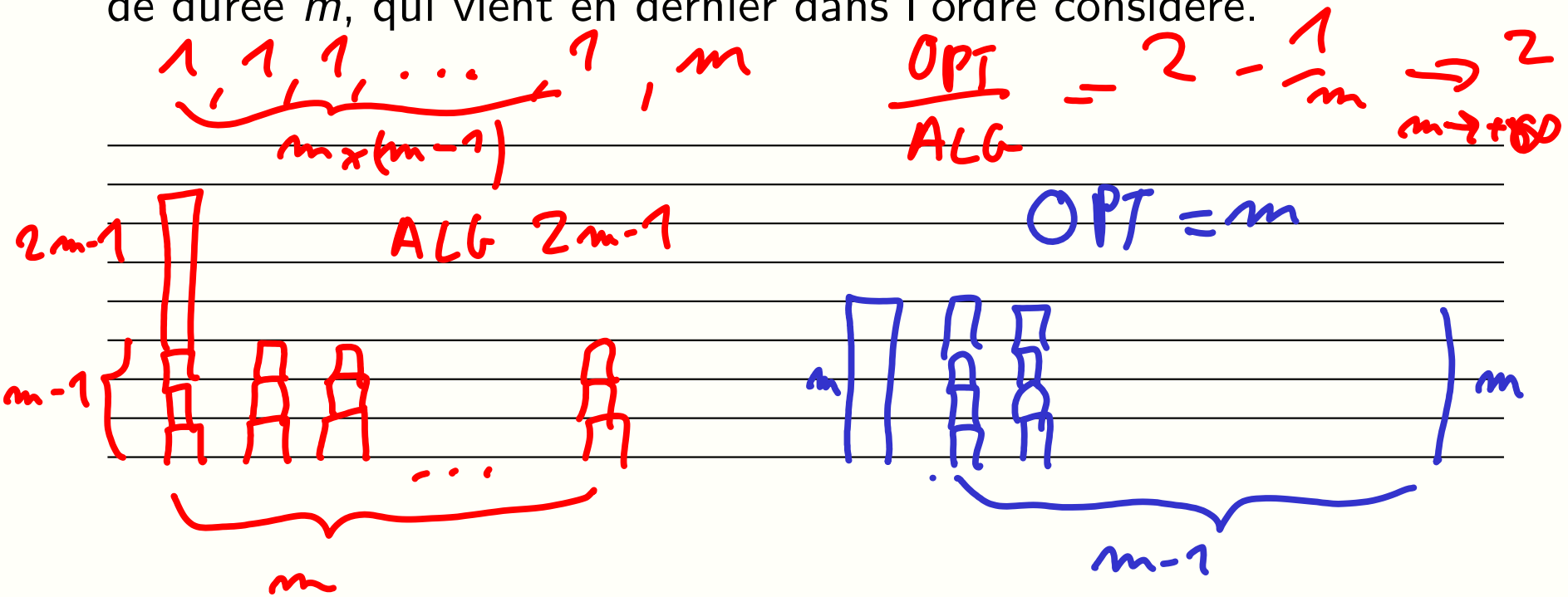
A-t-on prouve le meilleur ratio ?

- on a prouve $\rho \leq 2$, mais a-t-on $\rho < 2$?
- il faut trouver un exemple ou l'algo donne 2 ou bien s'approche aussi pres qu'on veut de 2

A-t-on prouve le meilleur ratio ?

- on a prouve $\rho \leq 2$, mais a-t-on $\rho < 2$?
- il faut trouver un exemple ou l'algo donne 2 ou bien s'approche aussi pres qu'on veut de 2

Exemple : m machines, $m(m-1)$ taches de duree 1 et une tache de duree m, qui vient en dernier dans l'ordre considere.



A-t-on prouve le meilleur ratio ?

- on a prouve $\rho \leq 2$, mais a-t-on $\rho < 2$?
- il faut trouver un exemple ou l'algo donne 2 ou bien s'approche aussi pres qu'on veut de 2

Exemple : m machines, $m(m-1)$ taches de duree 1 et une tache de duree m, qui vient en dernier dans l'ordre considere.
Par l'algo on obtient $2m-1$ alors qu'on peut faire m.
$2m-1/m \to 2$ quand $m \to +\infty$

A-t-on prouve le meilleur ratio ?

- on a prouve $\rho \leq 2$, mais a-t-on $\rho < 2$?
- il faut trouver un exemple ou l'algo donne 2 ou bien s'approche aussi pres qu'on veut de 2

Exemple : m machines, $m(m-1)$ taches de duree 1 et une tache de duree m, qui vient en dernier dans l'ordre considere.
Par l'algo on obtient $2m-1$ alors qu'on peut faire m.
$2m-1/m \to 2$ quand $m \to +\infty$

- conclusion : le ratio d'approx de cet algo est exactement 2

A-t-on prouve le meilleur ratio ?

- on a prouve $\rho \leq 2$, mais a-t-on $\rho < 2$?
- il faut trouver un exemple ou l'algo donne 2 ou bien s'approche aussi pres qu'on veut de 2

Exemple : m machines, $m(m-1)$ taches de duree 1 et une tache de duree m, qui vient en dernier dans l'ordre considere.
Par l'algo on obtient $2m-1$ alors qu'on peut faire m.
$2m-1/m \to 2$ quand $m \to +\infty$

- conclusion : le ratio d'approx de cet algo est exactement 2

- meilleur ratio avec un autre algo ?

Deuxieme algo : Longest Processing

ALGO :

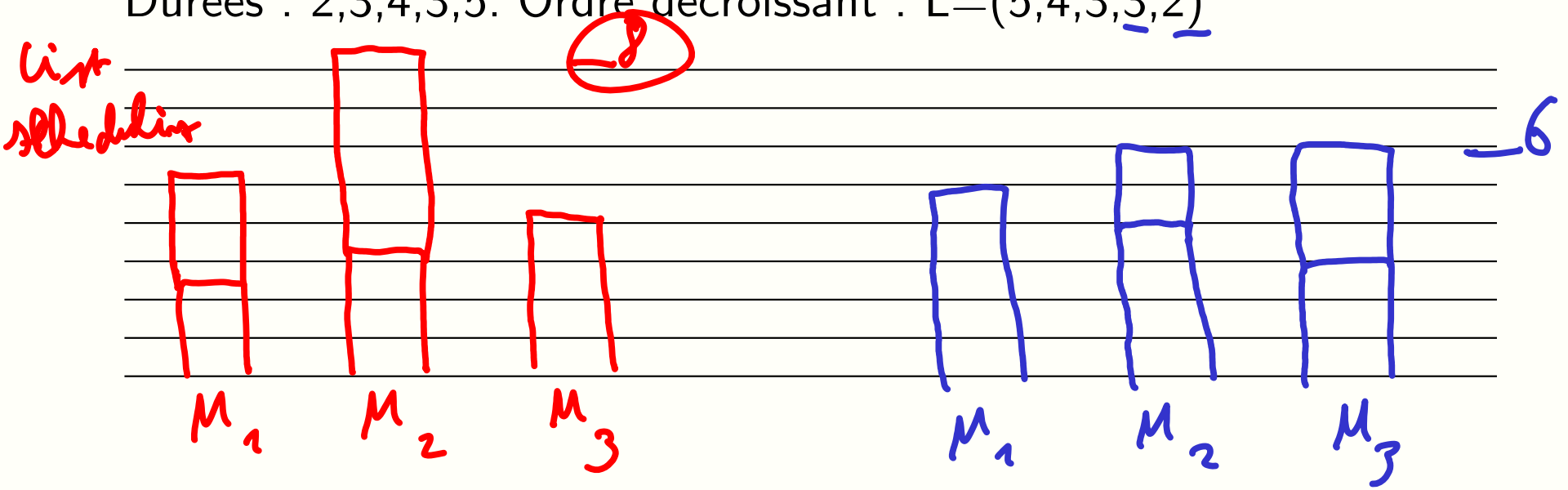
Meme algo que *List Scheduling* mais avec un ordre d'arrivee des taches bien choisi : par ordre decroissant des durees.

Deuxieme algo : Longest Processing

ALGO :

Meme algo que *List Scheduling* mais avec un ordre d'arrivee des taches bien choisi : par ordre decroissant des durees.

Durees : 2,3,4,3,5. Ordre decroissant : L=(5,4,3,3,2)



Deuxieme algo : Longest Processing

ALGO :

Meme algo que *List Scheduling* mais avec un ordre d'arrivee des taches bien choisi : par ordre decroissant des durees.

Durees : 2,3,4,3,5. Ordre decroissant : L=(5,4,3,3,2)

Analyse :

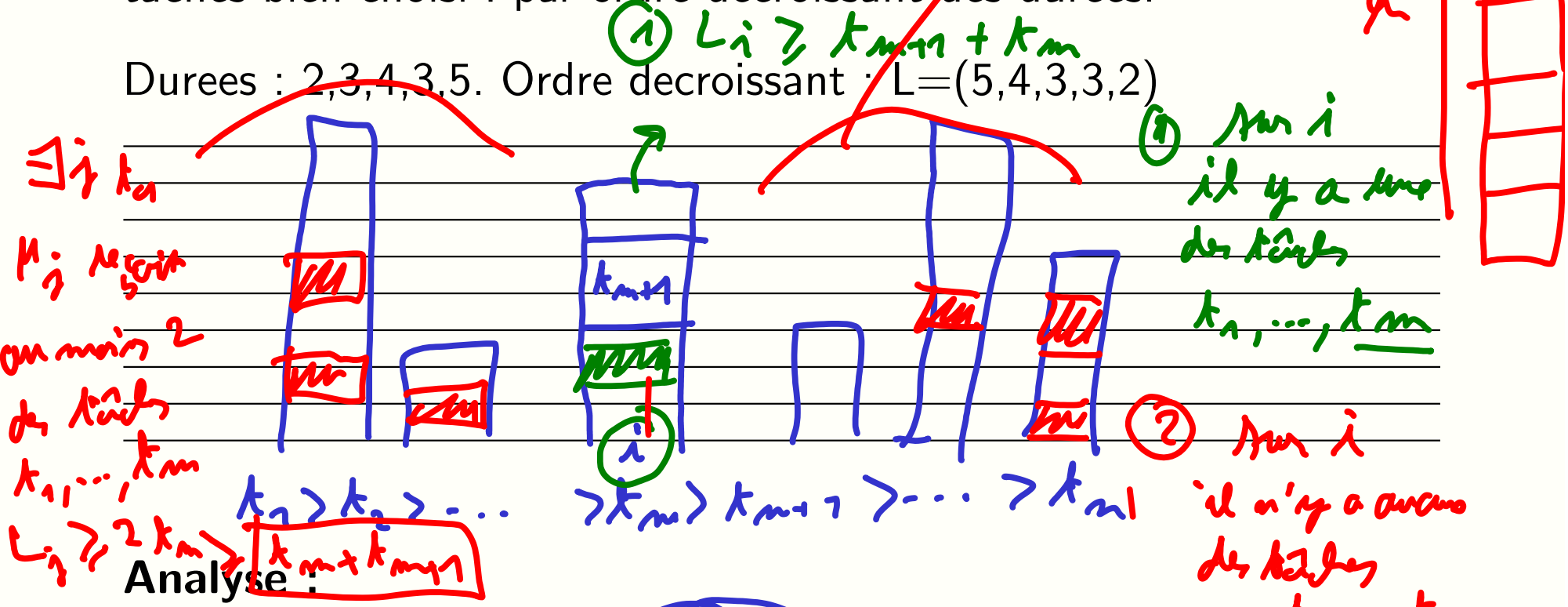
- on peut supposer que $n > m$, sinon l'algo trouve toujours l'optimal L^*

Deuxieme algo : Longest Processing

ALGO :

Meme algo que *List Scheduling* mais avec un ordre d'arrivee des taches bien choisi : par ordre decroissant des durees.

Durees : 2,3,4,3,5. Ordre decroissant : L=(5,4,3,3,2)



Analyse :

- on peut supposer que $n > m$, sinon l'algo trouve toujours l'optimal L^*
- dans toute solution optimale, chaque machine a une charge d'au moins $t_m + t_{m+1}$, d'ou $L^* \geq t_m + t_{m+1} \geq 2t_{m+1}$.

Deuxieme algo : Longest Processing

- d'apres l'analyse de l'algo precedent, on a $L_i \leq L^* + t_j$, on peut alors distinguer 2 cas :

Deuxieme algo : Longest Processing

- d'apres l'analyse de l'algo precedent, on a $L_i \leq L^* + t_j$, on peut alors distinguer 2 cas :
  - $j \leq m$: alors L_i n'a qu'une tache, c'est necessairement t_1 et $L^* = t_1$ est trouve par l'algo

Deuxieme algo : Longest Processing

- d'apres l'analyse de l'algo precedent, on a $L_i \leq L^* + t_j$, on peut alors distinguer 2 cas :
  - $j \leq m$: alors L_i n'a qu'une tache, c'est necessairement t_1 et $L^* = t_1$ est trouve par l'algo
  - $j \geq m+1$: alors $t_j \leq t_{m+1} \leq L^*/2$ et donc $L_i \leq 3/2 \cdot L^*$.

Deuxieme algo : Longest Processing

- d'apres l'analyse de l'algo precedent, on a $L_i \leq L^* + t_j$, on peut alors distinguer 2 cas :
  - $j \leq m$: alors L_i n'a qu'une tache, c'est necessairement t_1 et $L^* = t_1$ est trouve par l'algo
  - $j \geq m+1$: alors $t_j \leq t_{m+1} \leq L^*/2$ et donc $L_i \leq 3/2 \cdot L^*$.
- conclusion : $\rho \leq 3/2$

Deuxieme algo : Longest Processing

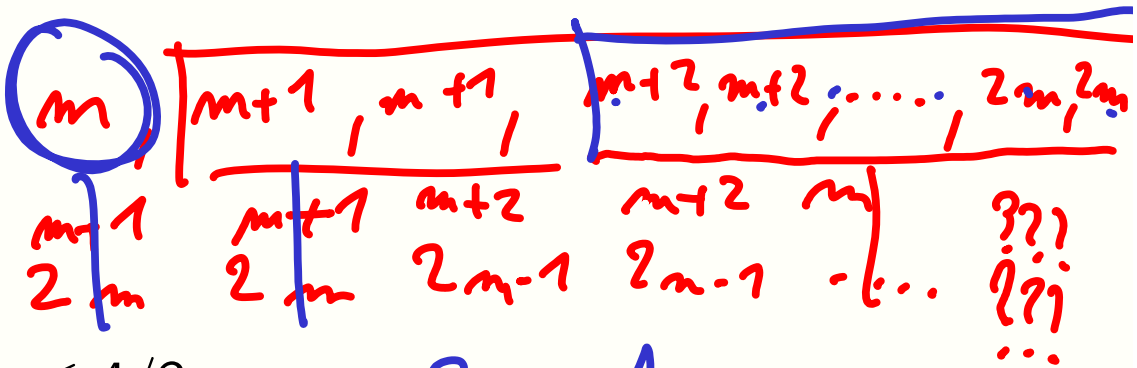
- d'apres l'analyse de l'algo precedent, on a $L_i \leq L^* + t_j$, on peut alors distinguer 2 cas :
  - $j \leq m$: alors L_i n'a qu'une tache, c'est necessairement t_1 et $L^* = t_1$ est trouve par l'algo
  - $j \geq m+1$: alors $t_j \leq t_{m+1} \leq L^*/2$ et donc $L_i \leq 3/2 \cdot L^*$.
- conclusion : $\rho \leq 3/2$
- A-t-on $\rho < 3/2$?

Deuxieme algo : Longest Processing

- d'apres l'analyse de l'algo precedent, on a $L_i \leq L^* + t_j$, on peut alors distinguer 2 cas :
  - $j \leq m$: alors L_i n'a qu'une tache, c'est necessairement t_1 et $L^* = t_1$ est trouve par l'algo
  - $j \geq m+1$: alors $t_j \leq t_{m+1} \leq L^*/2$ et donc $L_i \leq 3/2 \cdot L^*$.
- conclusion : $\rho \leq 3/2$
- A-t-on $\rho < 3/2$?
- Oui. On peut montrer $\rho \leq 4/3$.

Deuxieme algo : Longest Processing

- d'apres l'analyse de l'algo precedent, on a $L_i \leq L^* + t_j$, on peut alors distinguer 2 cas :
  - $j \leq m$: alors L_i n'a qu'une tache, c'est necessairement t_1 et $L^* = t_1$ est trouve par l'algo
  - $j \geq m+1$: alors $t_j \leq t_{m+1} \leq L^*/2$ et donc $L_i \leq 3/2 \cdot L^*$.



- conclusion : $\rho \leq 3/2$
- A-t-on $\rho < 3/2$?
- Oui. On peut montrer $\rho \leq 4/3$.
  Et c'est le mieux qu'on puisse montrer :
  m machines, $n = 2m+1$ taches, 2 taches de duree $m+1$, $m+2$, ..., $2m$ et une de duree m.