Chapitre 2 : Ordonnancement

ENSIIE - Module de Recherche Opérationnelle

Dimitri Watel (dimitri.watel@ensiie.fr)

2024

Problème d'atelier à 2 machines

Plusieurs points important :

- La machine M_1 ne traite les demandes que une par une
- La machine M_2 ne traite les demandes que une par une
- Les machines M_1 et M_2 peuvent travailler en parallèle
- Chaque ours doit être traité sur la machine M_1 avant la machine M_2.
- Selon la demande , les machines mettent plus ou moins de temps.

Problème d'atelier à 2 machines

On doit effectuer 8 tâches j_1 à j_8.
Voici un tableau indiquant combien de temps (en minute) met chaque machine pour travailler sur chaque tâche.

$t(j_i, M_j)$	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8
M_1	10	30	20	60	40	40	50	30
M_2	15	50	100	30	10	90	20	40

Question : comment effectuer les 8 tâches en un minimum de temps ?

L'algorithme de Johnson

$t(j_i, M_j)$	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8
M_1	10	30	20	60	40	40	50	30
M_2	15	50	100	30	10	90	20	40

- Partitionner l'ensemble des tâches en
  - $A = \{j_i, t(j_i, M_1) \leq t(j_i, M_2)\}$
  - $B = \{j_i, t(j_i, M_1) > t(j_i, M_2)\}$
- Ordonner A par ordre **croissant** de $t(j_i, M_1)$. On obtient une liste S_A.
- Ordonner B par ordre **décroissant** de $t(j_i, M_2)$ On obtient une liste S_B.
- Renvoyer la concaténation de S_A et S_B.

(voir l'exemple au tableau)

L'algorithme de Johnson

Propriété

L'algorithme de Johnson est optimal.

Par contre :

- Il ne fonctionne plus tel quel si on rajoute des contraintes (tel tâche doit être effectuée avant tel tâche ; la machine M_1 ne peut être utilisée plus de x minutes d'affilée, ...)
- Il ne fonctionne plus si on rajoute une machine, mais peut être adapté dans certains cas...

Si on supprime la contrainte d'ordre

Plusieurs points important :

- La machine M_1 ne traite les demandes que une par une
- La machine M_2 ne traite les demandes que une par une
- Les machines M_1 et M_2 peuvent travailler en parallèle
- ~~Chaque ours doit être traité sur la machine M_1 avant la machine M_2.~~ Un ours ne peut être traité sur la machine M_1 et la machine M_2 en même temps.
- Selon la demande , les machines mettent plus ou moins de temps.

Temps optimal

Pour simplifier, on note $a_i = t(j_i, M_1)$ et $b_i = t(j_i, M_2)$

Soit $T_1 = \sum_i a_i$.

Soit $T_2 = \sum_i b_i$.

Soit $M = \max_i a_i + b_i$.

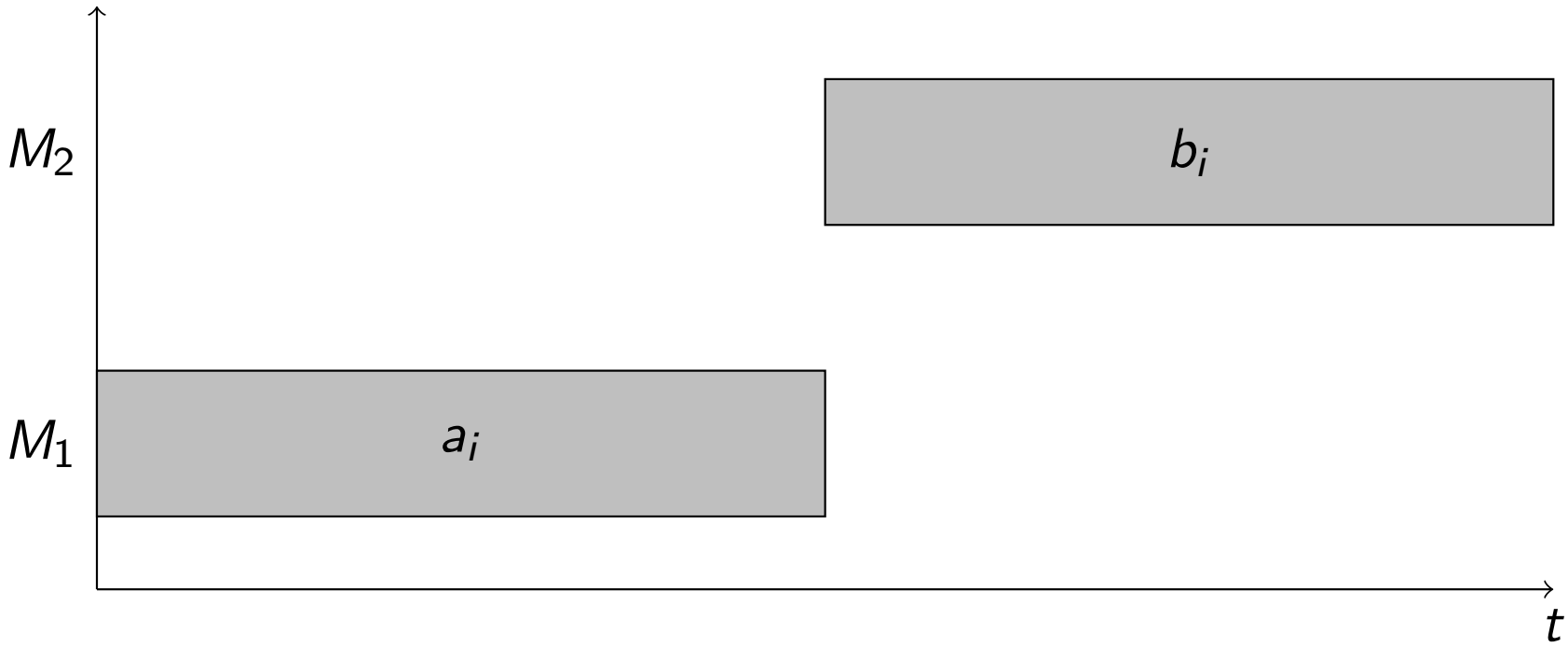
Théorème

Le temps optimal est $\max T_1, T_2, M$.

Comment atteindre ce temps ?

Cas 1 : $M \geq T_1, T_2$

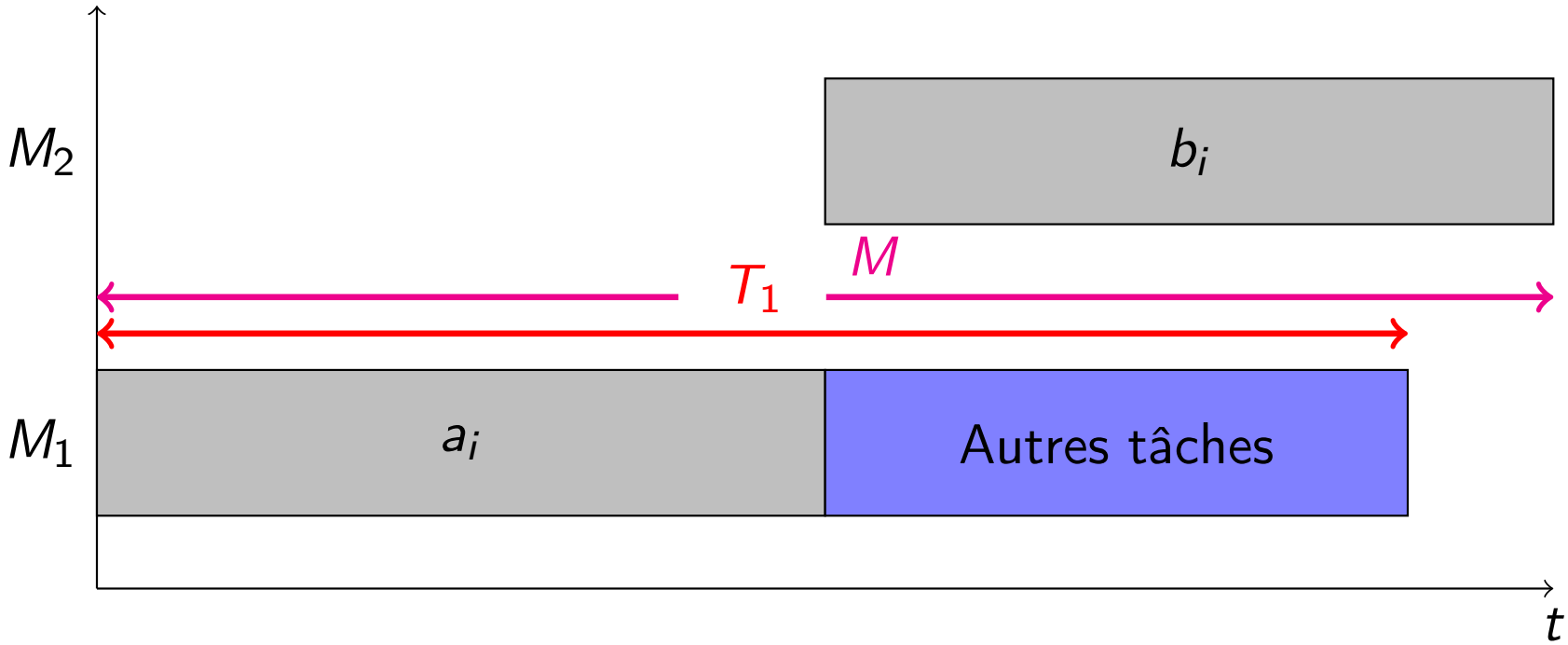
Soit i tel que $M = a_i + b_i$. Alors la solution suivante est optimale



1 : Placer la tâche i

Cas 1 : $M \geq T_1, T_2$

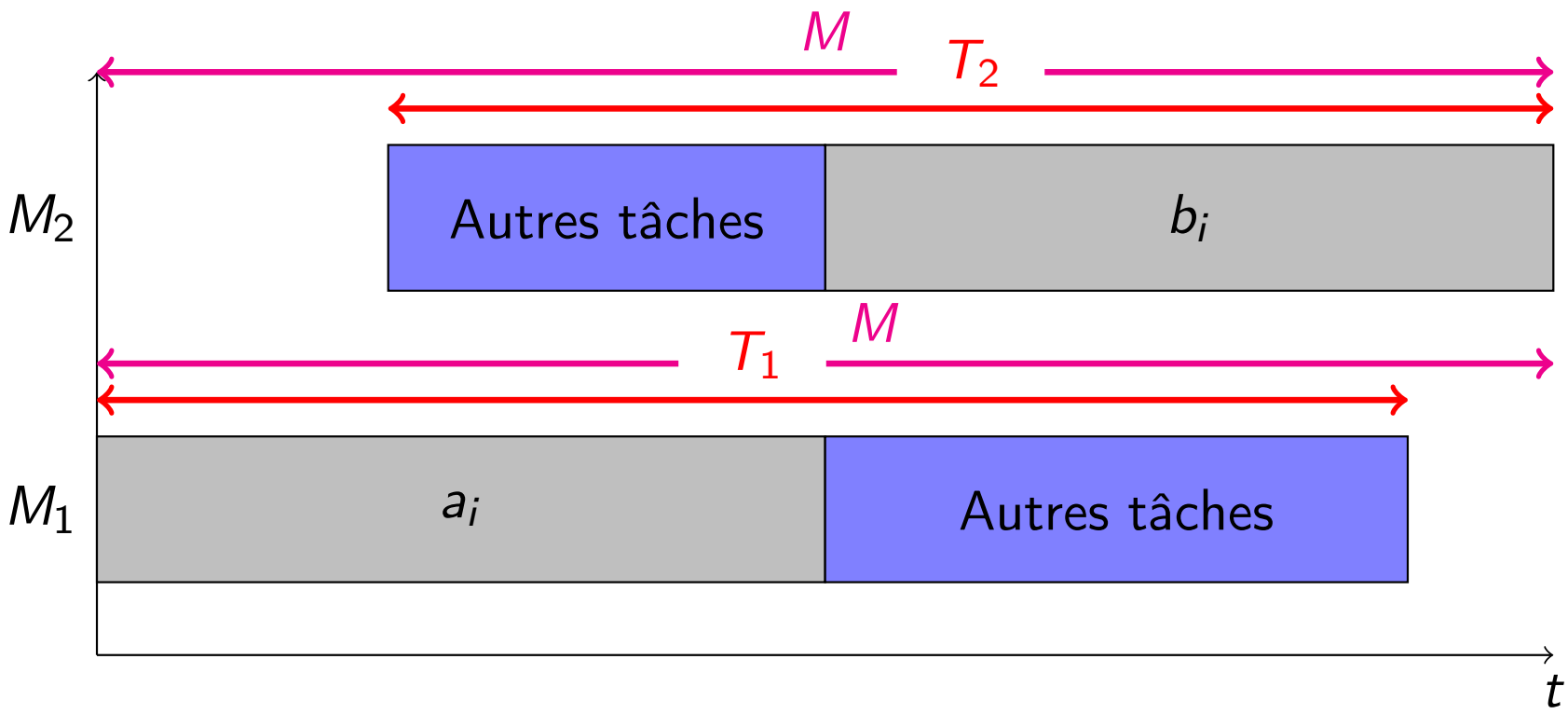
Soit i tel que $M = a_i + b_i$. Alors la solution suivante est optimale



2 : Il reste de la place pour les autres tâches sur M_1 car $M \geq T_1$

Cas 1 : $M \geq T_1, T_2$

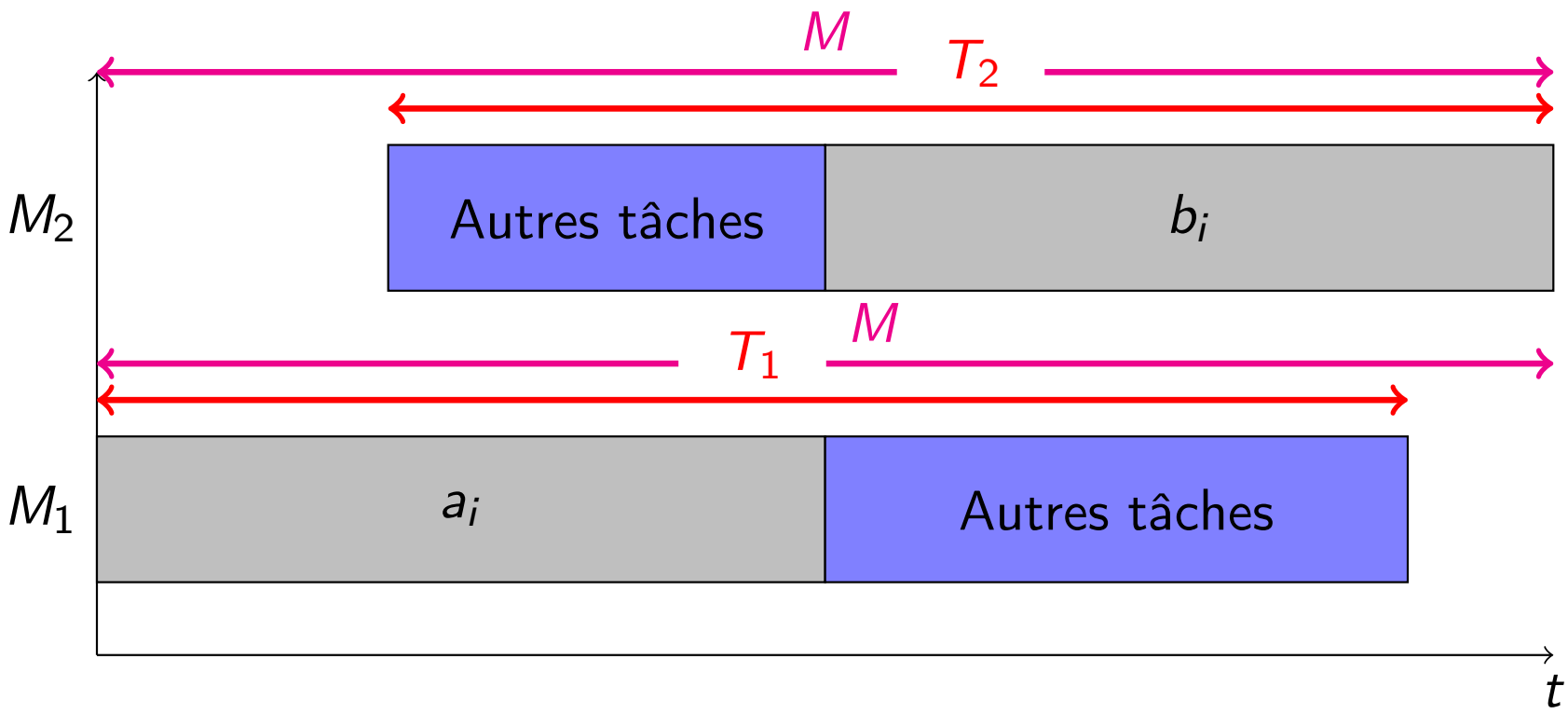
Soit i tel que $M = a_i + b_i$. Alors la solution suivante est optimale



3 : Il reste de la place pour les autres tâches sur M_2 car $M \geq T_2$

Cas 1 : $M \geq T_1, T_2$

Soit i tel que $M = a_i + b_i$. Alors la solution suivante est optimale



Aucune tâche n'est effectuée en même temps sur les 2 machines.

Cas 2 : $M < T_1$ ou T_2

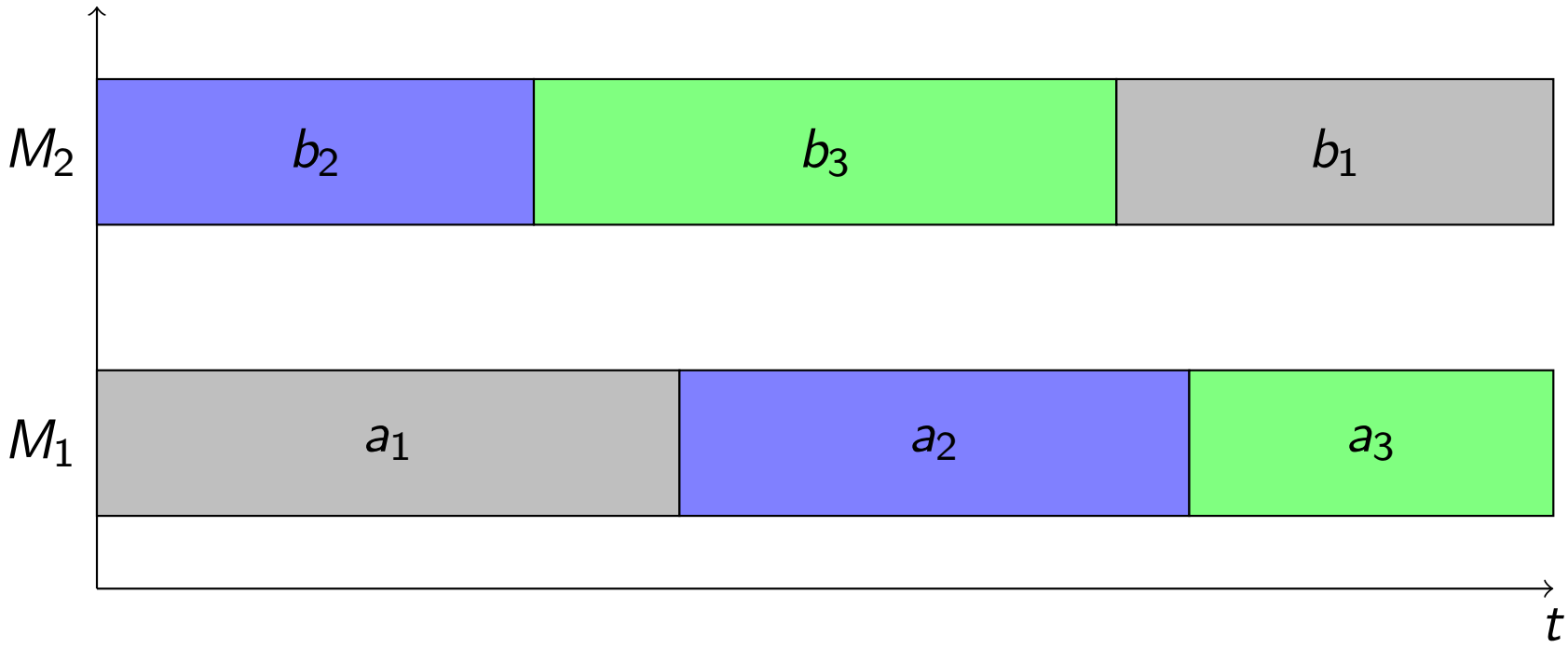
Ramenons nous à $T_1 = T_2$

Si $T_1 < T_2$, on ajoute $T_2 - T_1$ tâches fictives avec $a_i = 1$ et $b_i = 0$
Si $T_2 < T_1$, on ajoute $T_1 - T_2$ tâches fictives avec $a_i = 0$ et $b_i = 1$
Ainsi $T_1 = T_2$.

(Remarque : ces tâches peuvent être placées n'importe où sans provoquer de conflit.)

Cas 2.1 : $M < T_1 = T_2$ et il y a seulement 3 tâches

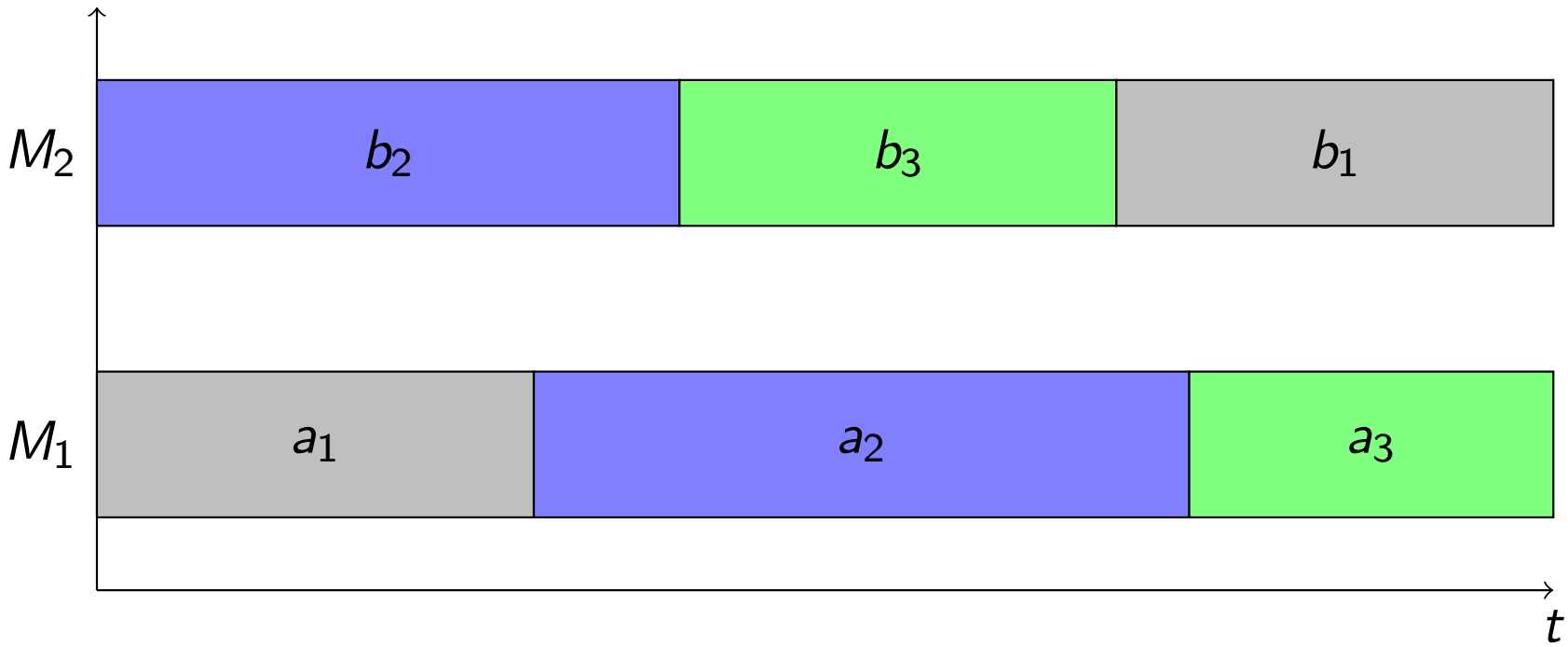
La procédure ci-dessous permet de trouver une solution optimale.



Si $a_1 \geq b_2$ et $b_1 \geq a_3$, la solution ci-dessus ne provoque aucun conflit.

Cas 2.1 : $M < T_1 = T_2$ et il y a seulement 3 tâches

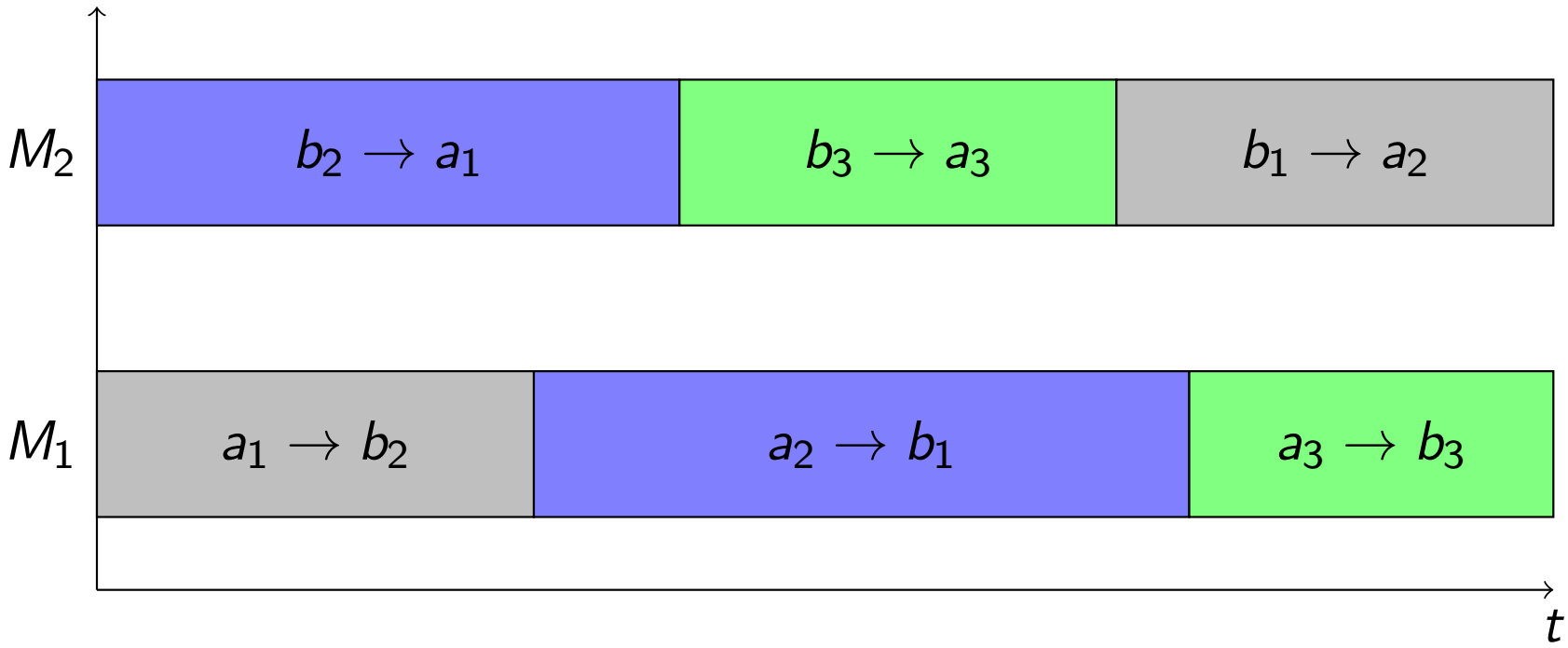
La procédure ci-dessous permet de trouver une solution optimale.



Si $a_1 < b_2$, il y a conflit (avec la tâche 2), mais alors ...

Cas 2.1 : $M < T_1 = T_2$ et il y a seulement 3 tâches

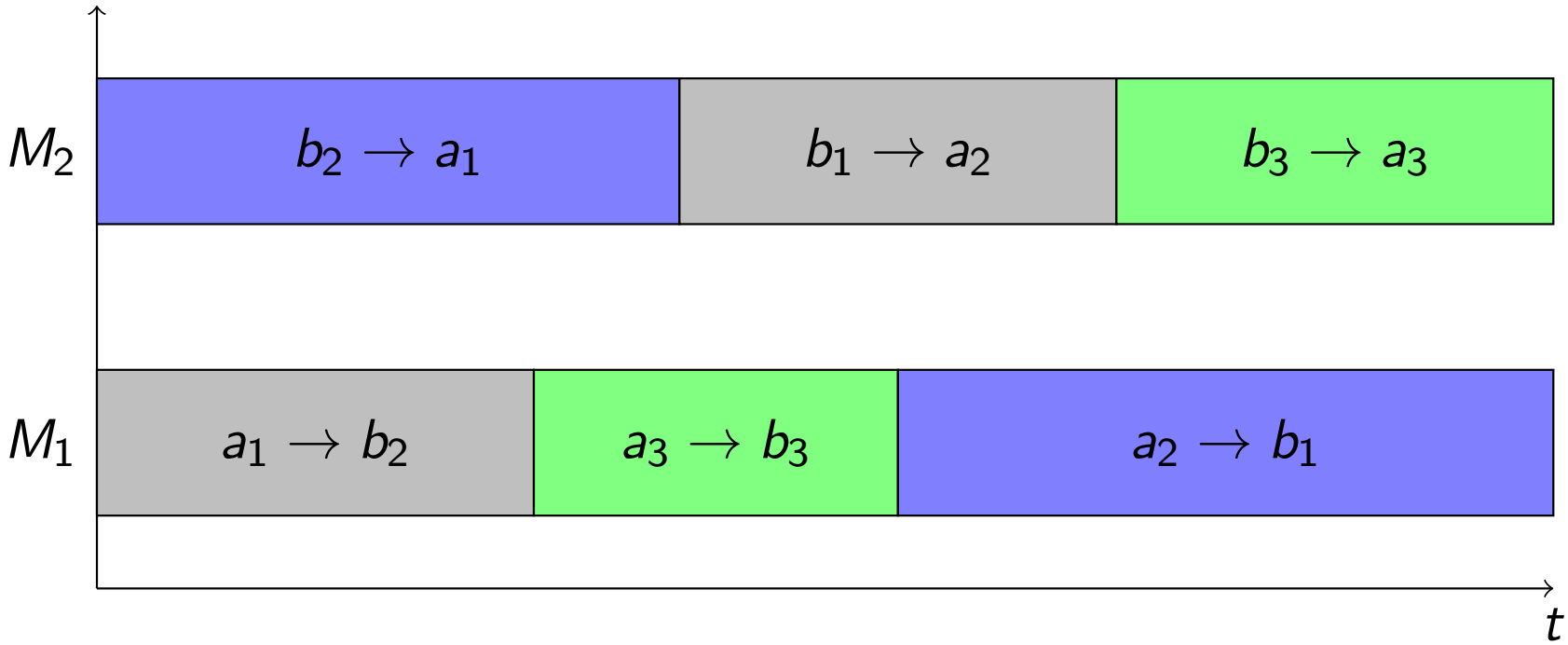
La procédure ci-dessous permet de trouver une solution optimale.



Si $a_1 < b_2$, il y a conflit (avec la tâche 2), mais alors ... renommons j_1 en j_2 et j_2 et en j_1 et M_1 en M_2 et M_2 en M_1. On a alors $a_1 \geq b_2$. (avec les nouvelles valeurs de a_1 et b_2)

Cas 2.1 : $M < T_1 = T_2$ et il y a seulement 3 tâches

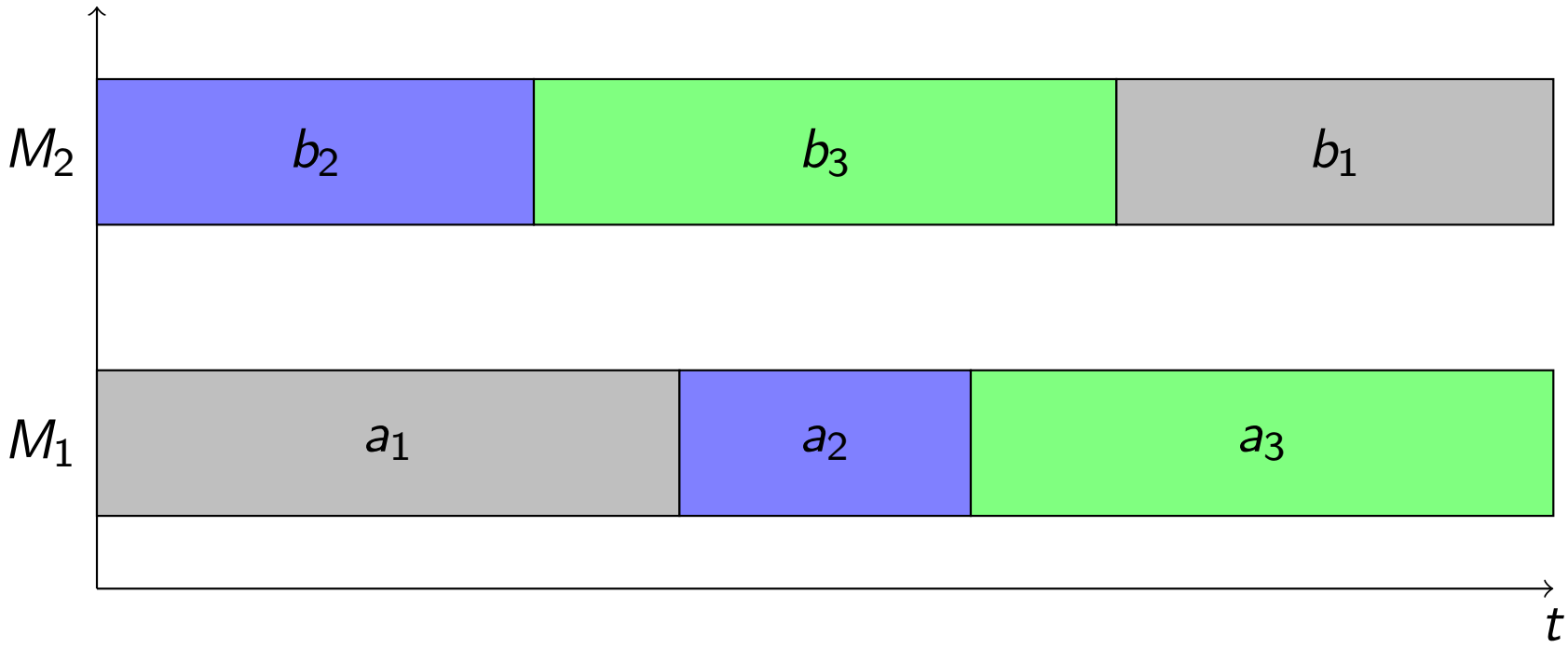
La procédure ci-dessous permet de trouver une solution optimale.



Si $a_1 < b_2$, il y a conflit (avec la tâche 2), mais alors ... renommons j_1 en j_2 et j_2 et en j_1 et M_1 en M_2 et M_2 en M_1. On a alors $a_1 \geq b_2$. Et on peut retenter la solution proposée au slide 13.

Cas 2.1 : $M < T_1 = T_2$ et il y a seulement 3 tâches

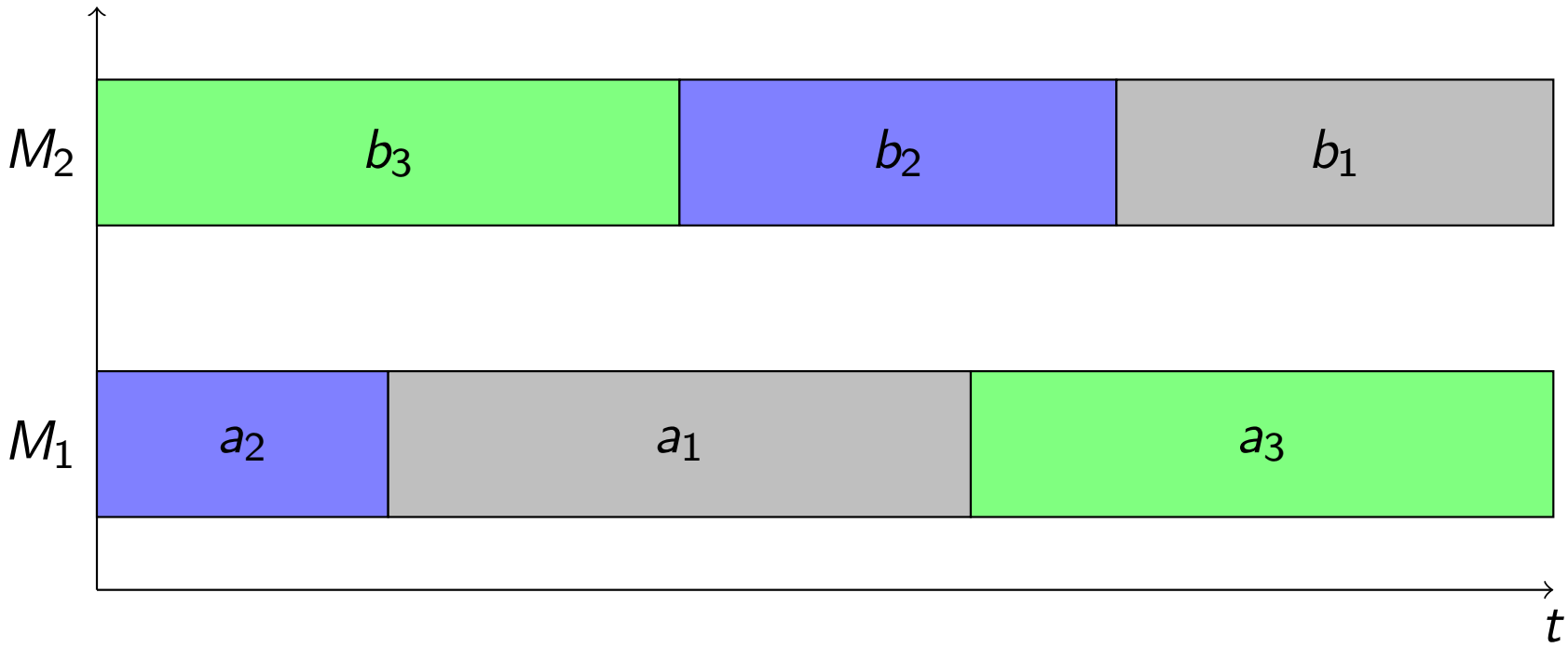
La procédure ci-dessous permet de trouver une solution optimale.



Si $a_1 \geq b_2$ mais $b_1 < a_3$, il y a conflit (avec la tâche 3), mais alors ...

Cas 2.1 : $M < T_1 = T_2$ et il y a seulement 3 tâches

La procédure ci-dessous permet de trouver une solution optimale.



Si $a_1 \geq b_2$ mais $b_1 < a_3$, il y a conflit (avec la tâche 3), mais alors ... la solution ci-dessus fonctionne car $a_1 + a_3 > b_1 + b_2$. (Rappel : $a_3 \leq b_1 + b_2$ car $M < T_1, T_2$)

Cas 2.2 : $M < T_1 = T_2$ et il y a $n > 3$ tâches

Astuce

On peut toujours se ramener à 3 tâches avec l'algorithme suivant :

Créer 3 ensembles J_1, J_2, J_3 vides
Pour k de 1 à n **Faire**
  Pour p de 1 à 3 **Faire**
    Si $\sum_{i \in J_p \cup \{k\}} a_i + b_i \leq T_1$ **Alors**
      Ajouter k à J_p

Théorème

À la fin, toutes les tâches sont dans J_1, J_2 ou J_3.

Cas 2.2 : $M < T_1 = T_2$ et il y a $n > 3$ tâches

L'algorithme est donc le suivant :

- construire les trois ensembles J_1, J_2 et J_3
- créer l'instance j'_1, j'_2 et j'_3 où $a'_i = \sum_{j_k \in J_i} a_k$ et $b'_i = \sum_{j_k \in J_i} b_k$
- trouver une solution avec l'algorithme du cas 2.1
- remplacer les occurrences de j'_1 par les tâches de J_1, dans n'importe quel ordre. De même pour j'_2 et j'_3.

A la fin il ne peut y avoir de conflit puisque cela impliquerait qu'il y a un conflit avec une des tâches $j_{1'}$, $j_{2'}$ ou $j_{3'}$.

Au programme, mais vu en TD

- Adapter PERT et/ou Potentiel Tâche (MPM) avec une contrainte supplémentaire.
- Algorithme de Johnson dans le cas de 3 machines M_1, M_2, M_3 telles que
  - $\max_i t(j_i, M_2) \leq \min_i t(j_i, M_1)$
  - $\max_i t(j_i, M_2) \leq \min_i t(j_i, M_3)$.