Exact algorithms for the bilevel scheduling of uniform parallel machines in the context of coupling maintenance and scheduling decisions

Quentin Schau[1,2], Olivier Ploton[1], Vincent T'kindt[1], Federico Della Croce[2], Han Hoogeveen[3]

[1]University of Tours,
LIFAT, France.

[2]DIGEP, Politecnico di Torino,
CNR, IEIIT, Torino, Italy.

[3]Utrecht University,
Department of Information and Computing Sciences, Netherlands.

Outline

Bilevel scheduling: problem definition

Context

We want to maximize the production and the system health

Bilevel scheduling: problem definition

Context

We want to maximize the production and the system health

We are interested in coupling maintenance and scheduling decisions.
So we are considering **2 speeds for machines**

Bilevel scheduling: problem definition

Context

We want to maximize the production and the system health

We are interested in coupling maintenance and scheduling decisions.
So we are considering **2 speeds for machines**

- if maintenance is required $\Rightarrow$ set speed to V_0

Bilevel scheduling: problem definition

Context

We want to maximize the production and the system health

We are interested in coupling maintenance and scheduling decisions.
So we are considering **2 speeds for machines**

- if maintenance is required $\Rightarrow$ set speed to V_0
- else set speed to V_1.

Bilevel scheduling: problem definition

Context

We want to maximize the production and the system health

We are interested in coupling maintenance and scheduling decisions.
So we are considering **2 speeds for machines**

- if maintenance is required $\Rightarrow$ set speed to V_0
- else set speed to V_1.

Global scheduler

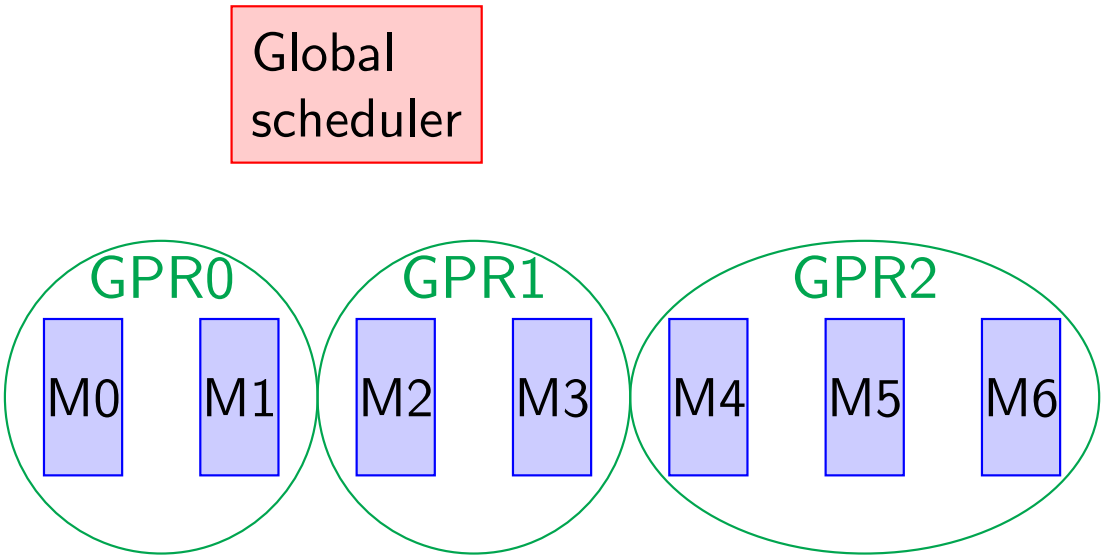
Bilevel scheduling: problem definition

Context

We want to maximize the production and the system health

We are interested in coupling maintenance and scheduling decisions.
So we are considering **2 speeds for machines**

- if maintenance is required $\Rightarrow$ set speed to V_0
- else set speed to V_1.



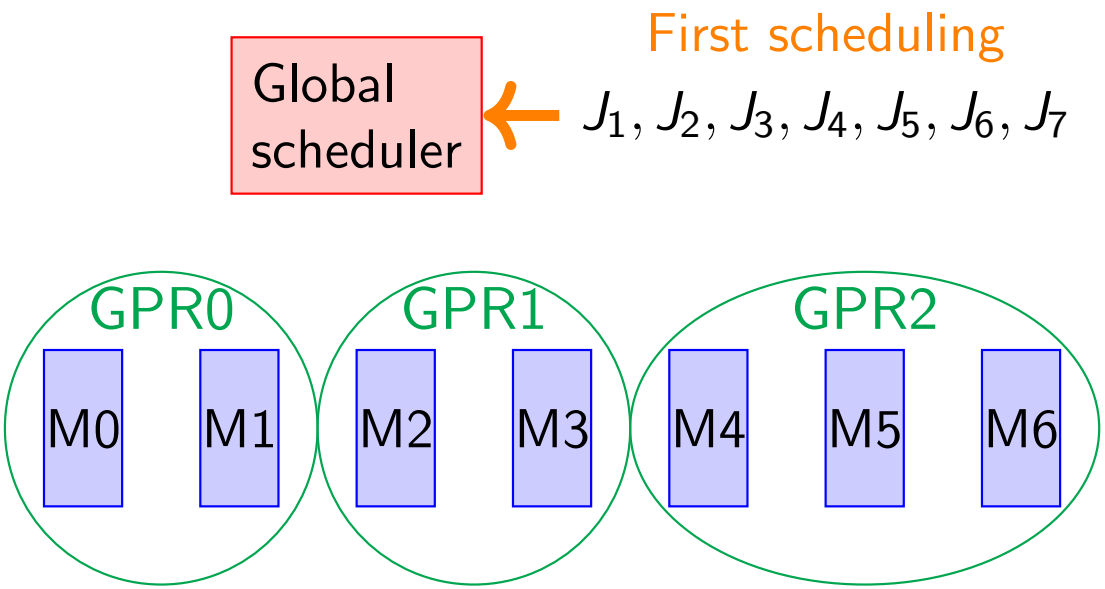
Bilevel scheduling: problem definition

Context

We want to maximize the production and the system health

We are interested in coupling maintenance and scheduling decisions.
So we are considering **2 speeds for machines**

- if maintenance is required $\Rightarrow$ set speed to V_0
- else set speed to V_1.



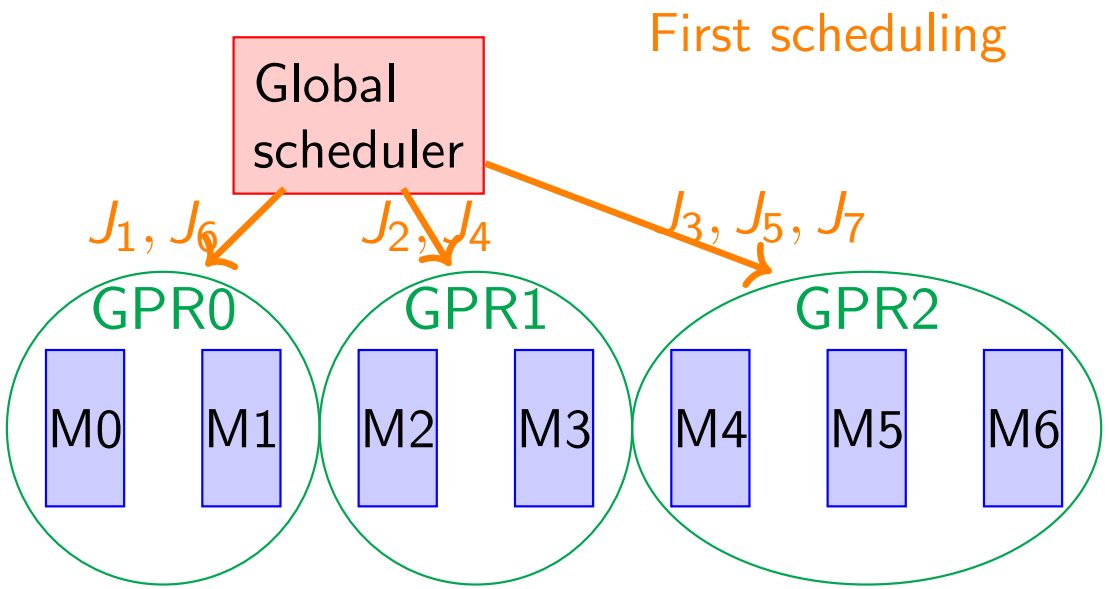
Bilevel scheduling: problem definition

Context

We want to maximize the production and the system health

We are interested in coupling maintenance and scheduling decisions.
So we are considering **2 speeds for machines**

- if maintenance is required $\Rightarrow$ set speed to V_0
- else set speed to V_1.



Bilevel scheduling: problem definition

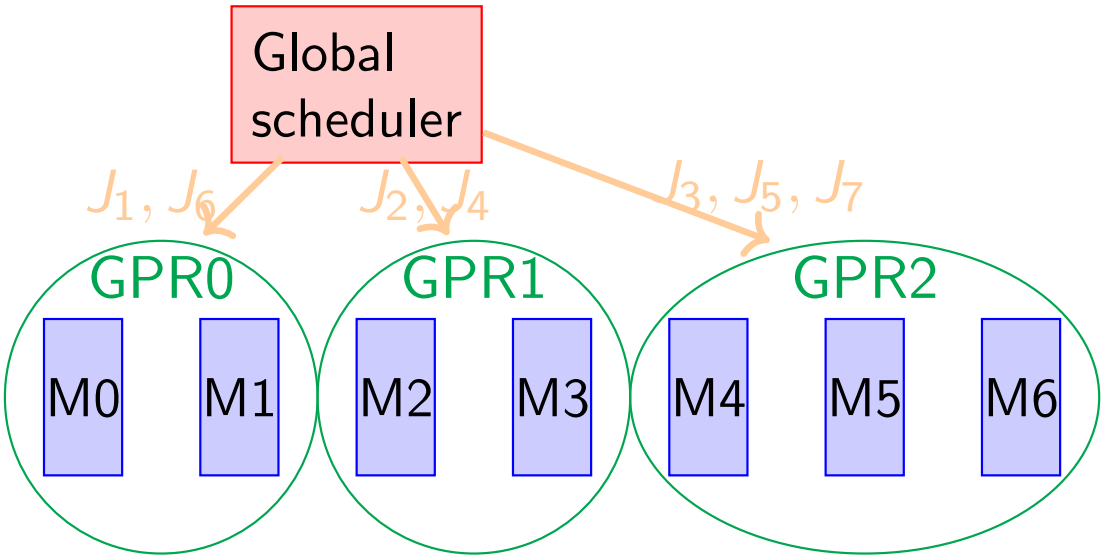
Context

We want to maximize the production and the system health

We are interested in coupling maintenance and scheduling decisions.
So we are considering **2 speeds for machines**

- if maintenance is required $\Rightarrow$ set speed to V_0
- else set speed to V_1.

After period T :



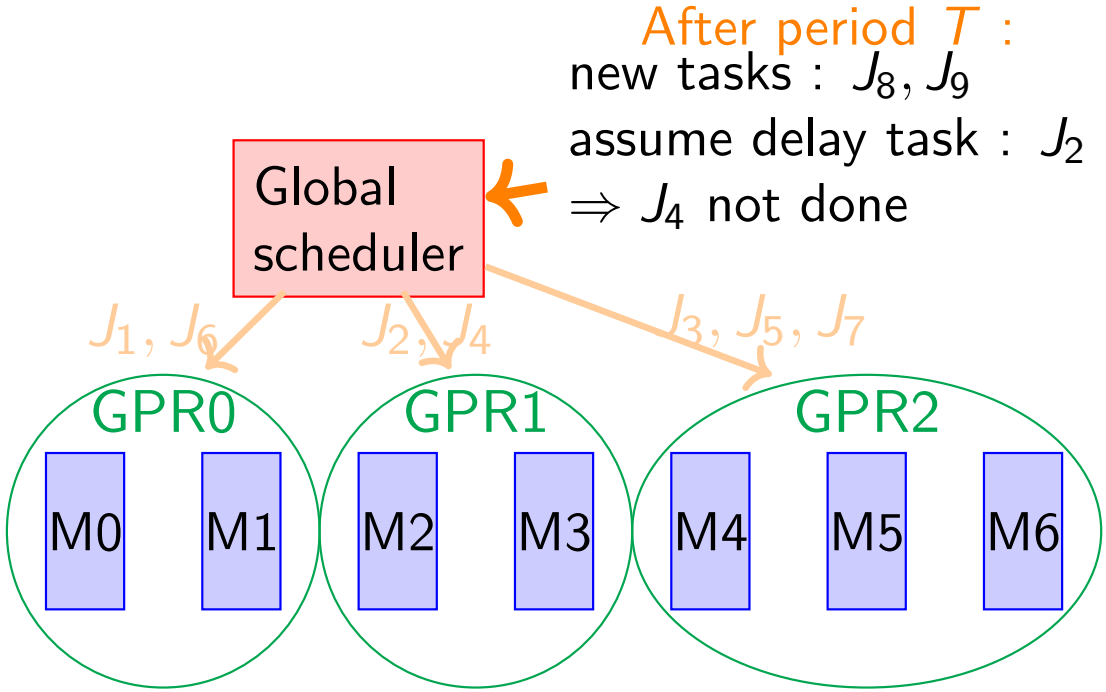
Bilevel scheduling: problem definition

Context

We want to maximize the production and the system health

We are interested in coupling maintenance and scheduling decisions.
So we are considering **2 speeds for machines**

- if maintenance is required $\Rightarrow$ set speed to V_0
- else set speed to V_1.



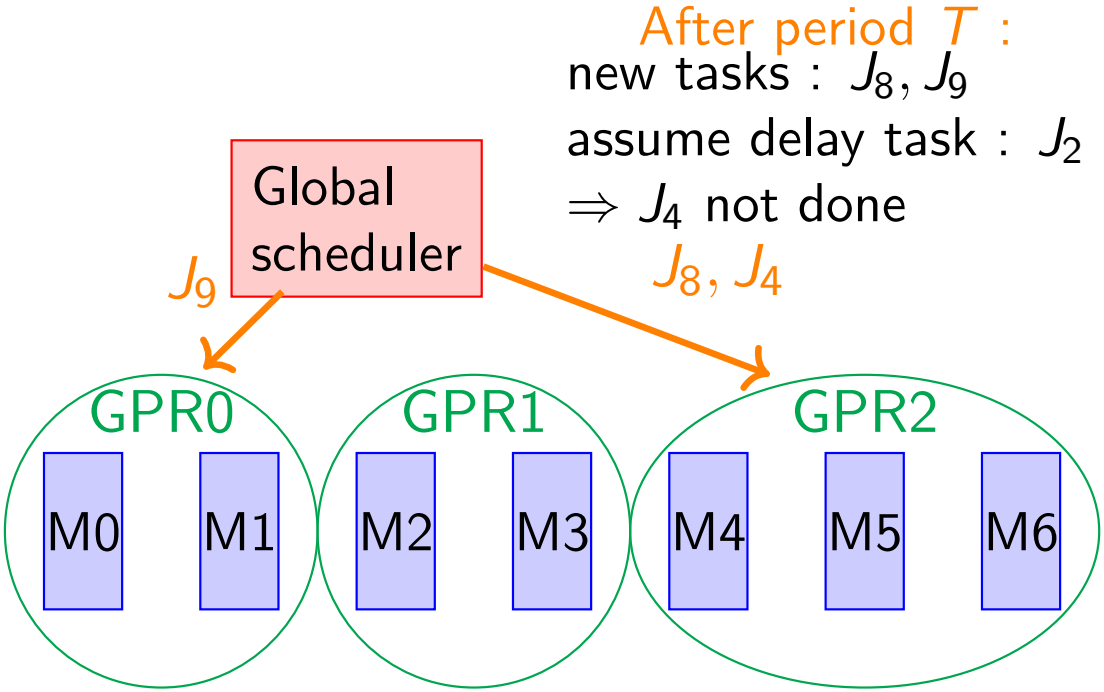
Bilevel scheduling: problem definition

Context

We want to maximize the production and the system health

We are interested in coupling maintenance and scheduling decisions.
So we are considering **2 speeds for machines**

- if maintenance is required $\Rightarrow$ set speed to V_0
- else set speed to V_1.



Bilevel scheduling: problem definition

Context

We focus on period T

Bilevel scheduling: problem definition

Context

We focus on period T

Leader

We have 2 agents:

- A leader

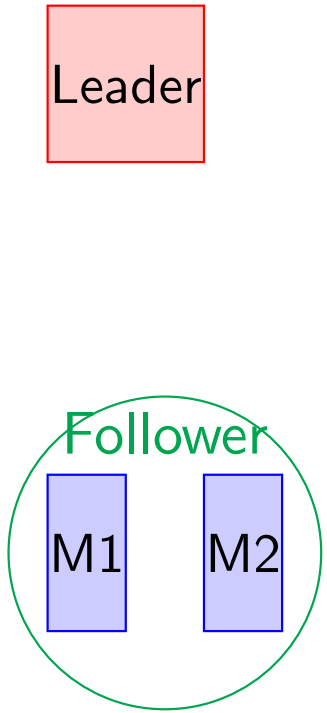
Bilevel scheduling: problem definition

Context

We focus on period T

We have 2 agents:

- A leader
- A follower composed of several machines



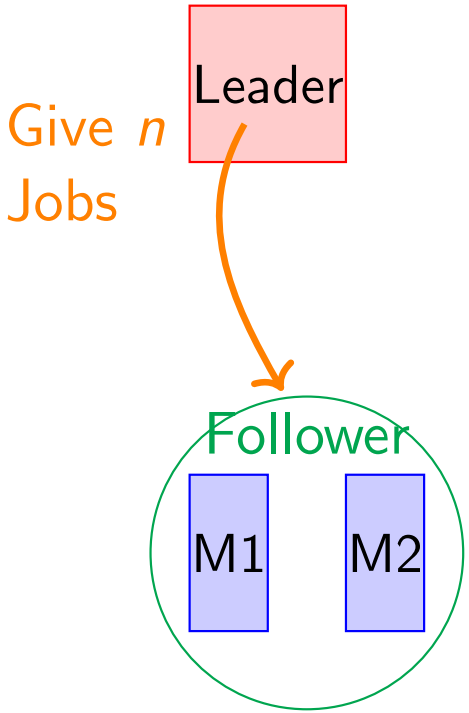
Bilevel scheduling: problem definition

Context

We focus on period T

We have 2 agents:

- A leader **ONLY selects** n jobs among $N \rightarrow$ minimize $\sum_j w_j U_j^L$
- A follower composed of several machines



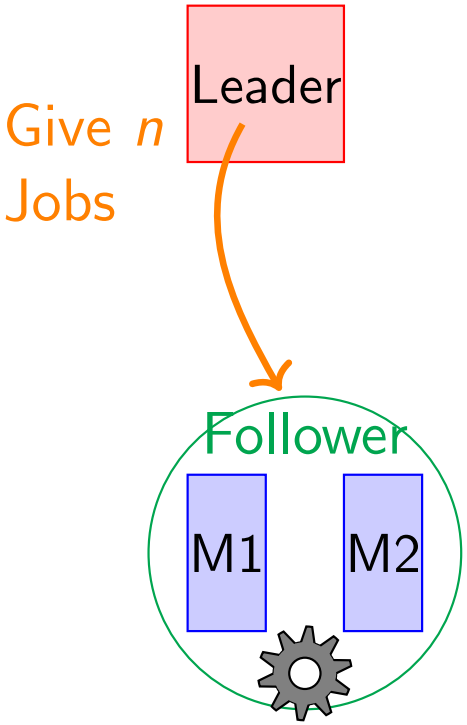
Bilevel scheduling: problem definition

Context

We focus on period T

We have 2 agents:

- A leader **ONLY selects** n jobs among $N \rightarrow$ minimize $\sum_j w_j U_j^L$
- A follower composed of several machines **schedules** jobs $\rightarrow$ minimize $\sum_j C_j^F$



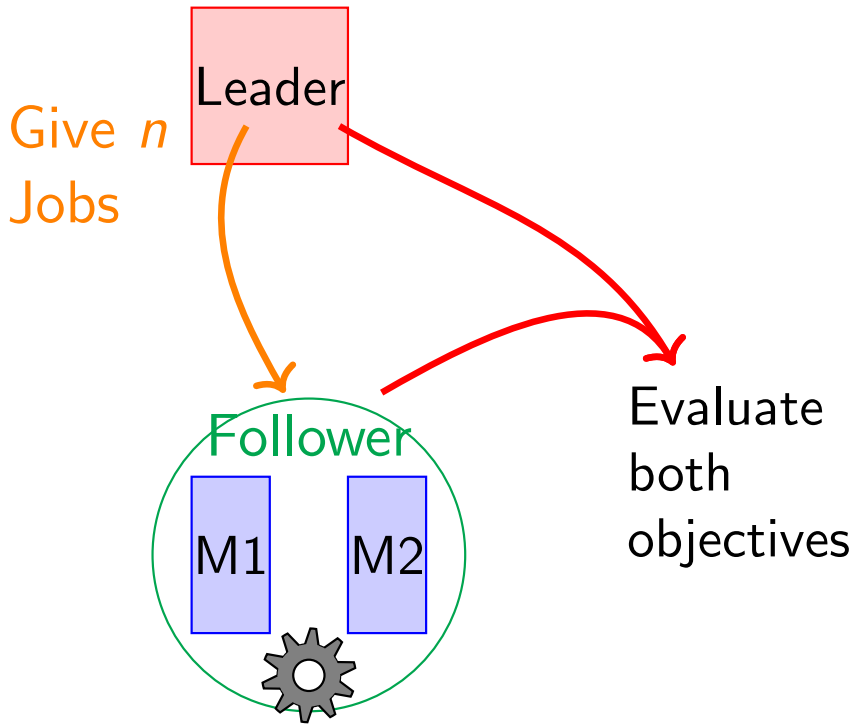
Bilevel scheduling: problem definition

Context

We focus on period T

We have 2 agents:

- A leader **ONLY selects** n jobs among $N \rightarrow$ minimize $\sum_j w_j U_j^L$
- A follower composed of several machines **schedules** jobs $\rightarrow$ minimize $\sum_j C_j^F$



Bilevel scheduling: problem definition

Definition

We model our problem as a bilevel scheduling problem

Bilevel scheduling: problem definition

Definition

We model our problem as a bilevel scheduling problem

We consider the optimistic case, i.e.,

The follower returns the schedule $\rightarrow \min \sum_j w_j U_j^L$ **among all optimal schedules** for $\sum_j C_j^F$.

Bilevel scheduling: problem definition

Definition

We model our problem as a bilevel scheduling problem

We consider the optimistic case, i.e.,

The follower returns the schedule $\rightarrow \min \sum_j w_j U_j^L$ **among all optimal schedules** for $\sum_j C_j^F$.

$\Leftrightarrow Lex\left(\sum_j C_j^F, \sum_j w_j U_j^L\right)$ for the follower.

Bilevel scheduling: problem definition

Definition

We can define our problem as:

$$\min_{\substack{\mathcal{I}\subset\mathcal{J}\\|\mathcal{I}|=n}} \sum_{j\in\sigma^*} w_j U_j^L$$

$$\text{st.} \quad \sigma^* = \arg\min_{\sigma\in\mathfrak{S}_\mathcal{I}} \left(Lex \left(\sum_{j\in\sigma} C_j^F, \sum_{j\in\sigma} w_j U_j^L \right) \right) \tag{1}$$

where:

- $\mathfrak{S}_\mathcal{I}$ is the set of schedules formed of all jobs in $\mathcal{I}$.
- C_j^F is the completion time of the job j.
- $U_j^L = \begin{cases} 1 \text{ if } C_j^F > d_j \\ 0 \text{ otherwise} \end{cases}$

1. Introduction
   - Problem definition
   - Block structure property
   - Complexity by reduction

Bilevel scheduling: block structure property

Property

Theorem $Q||\sum C_j$ [1]

$Q||\sum C_j$ is polynomial in $\mathcal{O}(n\ log(n))$

Assignment problem: **larger** processing times $p_1 \geqslant p_2 \geqslant \ldots \geqslant p_n$ are assigned to the **smaller** coefficients $t_j = \frac{k}{V_i}$.

Here, job j will be in the k-th position (counting from the end) on machine i.

[1] R. W. Conway, W. Maxwell, and L. Miller, *Theory of Scheduling*. Addison-Wesley Publishing Company, 1967

Bilevel scheduling: block structure property

Property

Theorem $Q||\sum C_j$ [1]

$Q||\sum C_j$ is polynomial in $\mathcal{O}(n\ log(n))$

Assignment problem: **larger** processing times $p_1 \geqslant p_2 \geqslant \ldots \geqslant p_n$ are assigned to the **smaller** coefficients $t_j = \frac{k}{V_i}$.

Here, job j will be in the k-th position (counting from the end) on machine i.

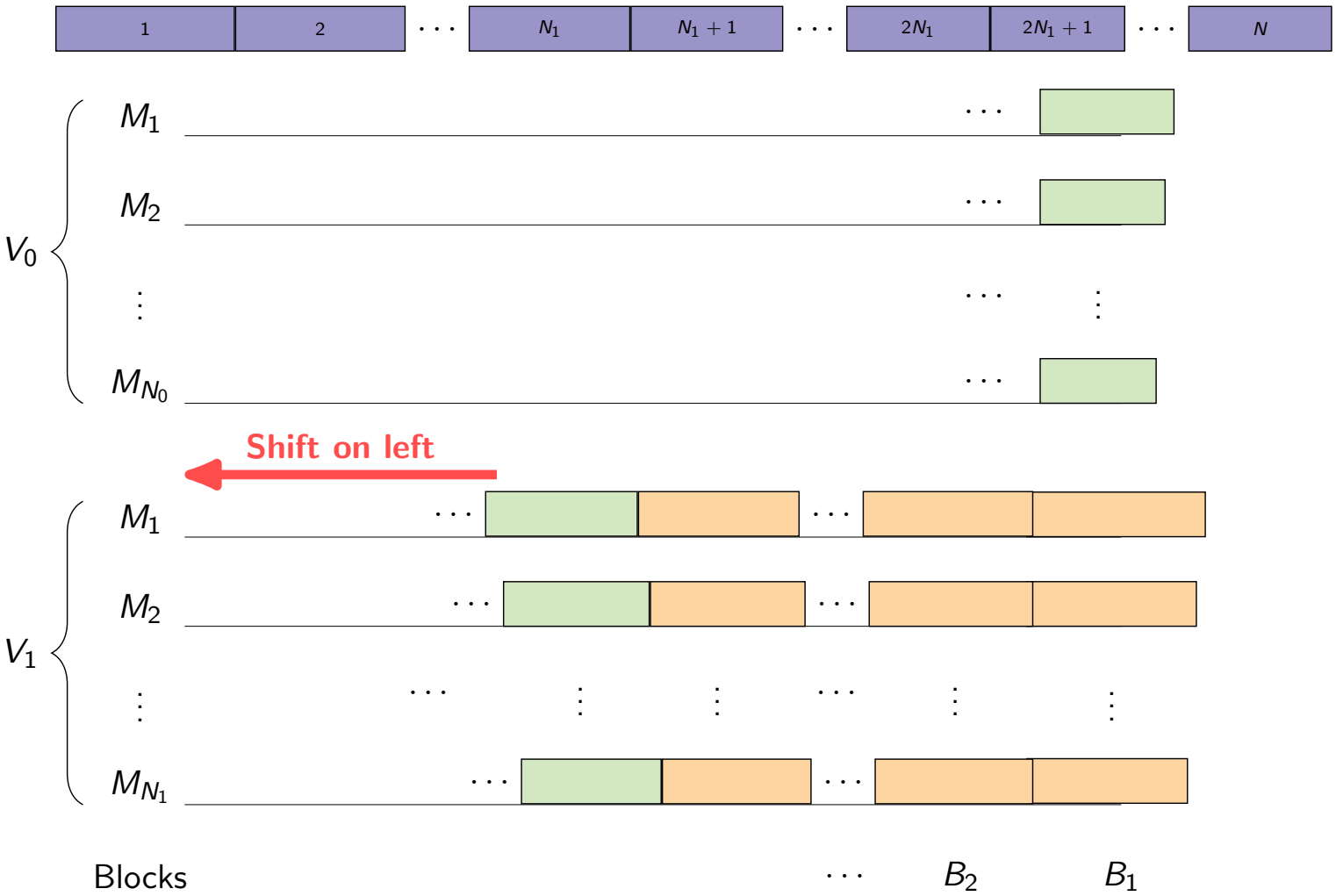
We focus on a special case with 2 speeds: $Q|V_i \in \{V_0, V_1\}|\sum_j C_j$

[1] R. W. Conway, W. Maxwell, and L. Miller, *Theory of Scheduling*. Addison-Wesley Publishing Company, 1967

Bilevel scheduling: block structure property

Properties

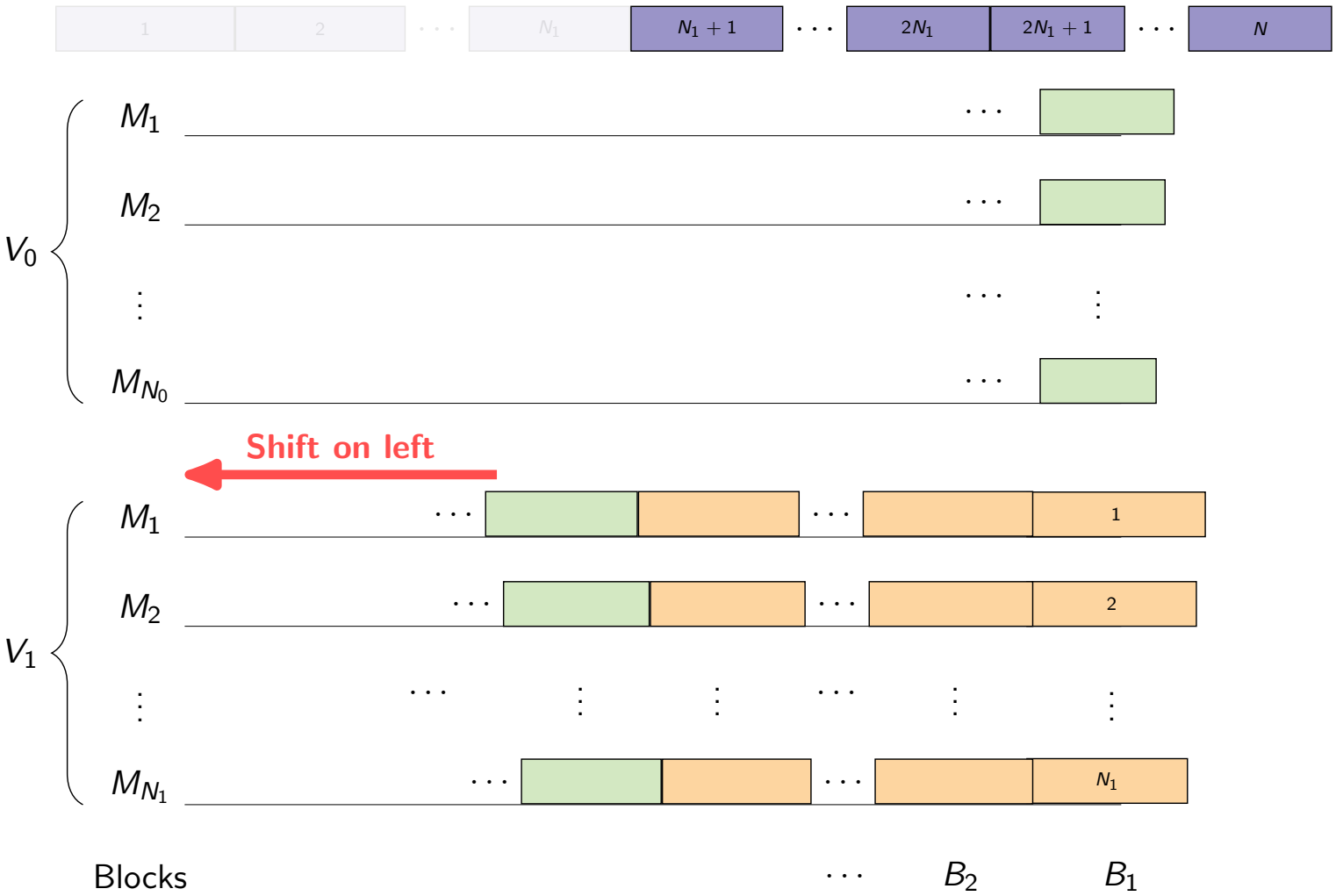
if V_0 divide V_1 n jobs sorted according LPT



Bilevel scheduling: block structure property

Properties

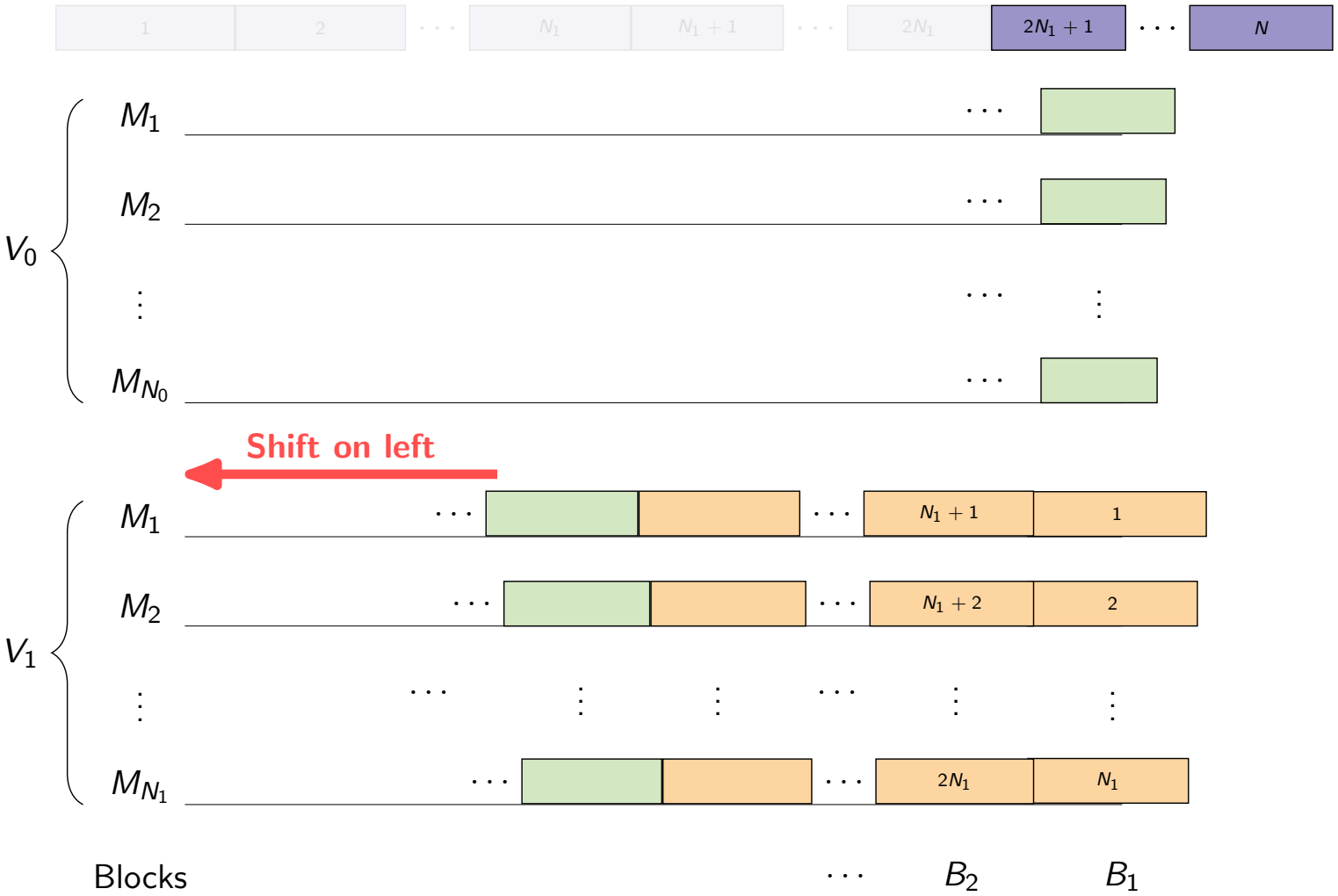
if V_0 divide V_1 n jobs sorted according LPT



Bilevel scheduling: block structure property

Properties

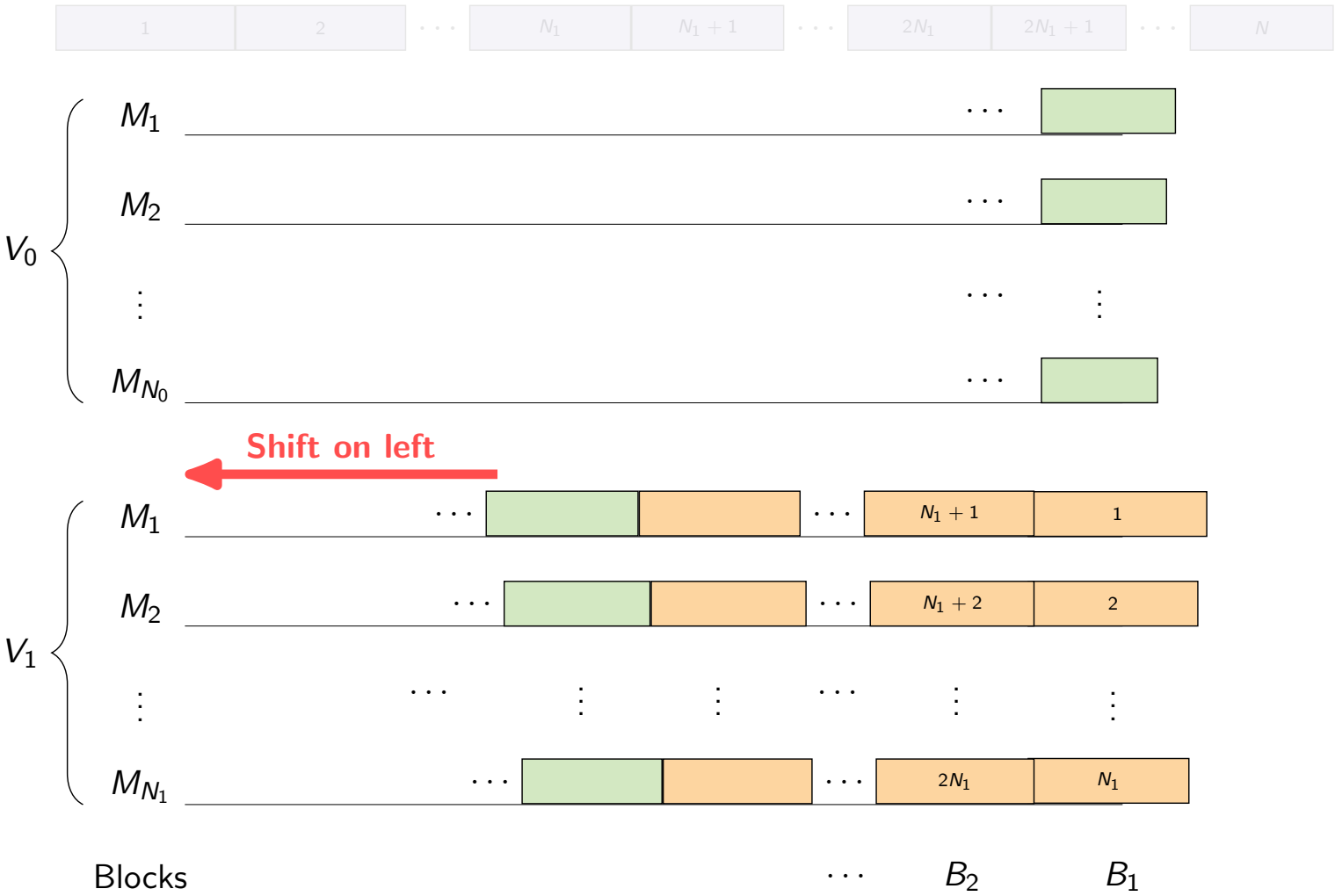
if V_0 divide V_1 n jobs sorted according LPT



Bilevel scheduling: block structure property

Properties

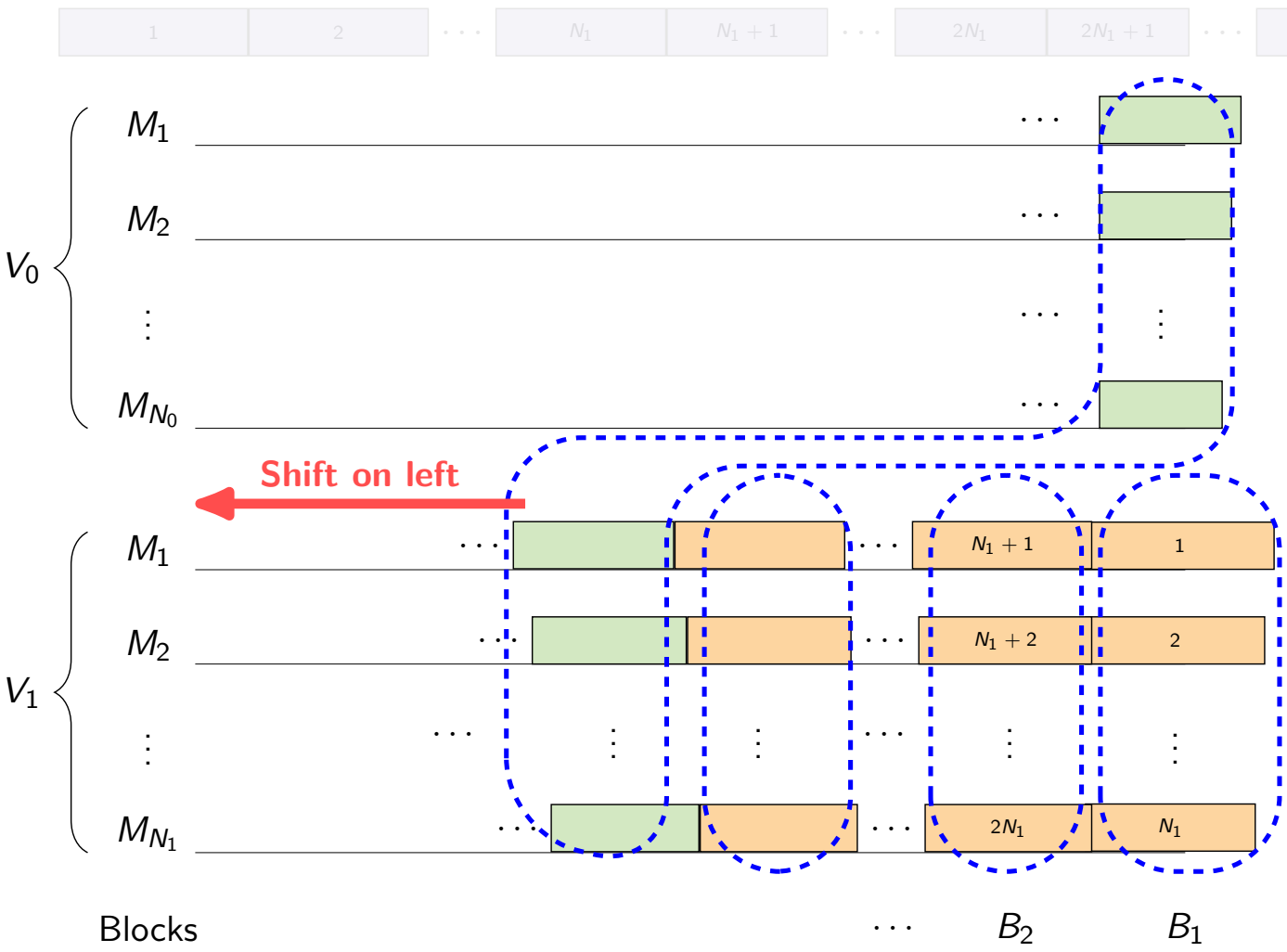
if V_0 divide V_1 n jobs sorted according LPT



Bilevel scheduling: block structure property

Properties

if V_0 divide V_1 n jobs sorted according LPT



- Applying a permutation in a block does not change the objective value

1. Introduction
   - Problem definition
   - Block structure property
   - Complexity by reduction

Bilevel scheduling: complexity by reduction

Complexity

Bilevel optimisation problems are difficult in terms of complexity .
What happens for our problem ?

Bilevel scheduling: complexity by reduction

Complexity

Bilevel optimisation problems are difficult in terms of complexity .
What happens for our problem ?

Theorem

The problem $Q|V_i \in \{V_0, V_1\}|Lex\left(\sum_j C_j^F, \sum_j w_j U_j^L\right)$ is $\mathcal{NP}$-hard in the strong sense by reduction from the numerical 3-dimensional matching ($NUM-3DM$).

Proof omitted

2 Exact methods

- MIP
- Column generation

2 Exact methods

- MIP
- Column generation

Exact methods

MIP formulation

Variables:

- $x_{ijk} \in \{0, 1\}$: $x_{ijk}=1 \Rightarrow$job j is selected & scheduled on location (i, k)

Exact methods

MIP formulation

Variables:

- $x_{ijk} \in \{0, 1\}$: $x_{ijk}=1 \Rightarrow$job j is selected & scheduled on location (i, k)
- $C_{ik} \in \mathbb{R}^+$: compl. time of the job selected & scheduled on loc. (i, k)

Exact methods

MIP formulation

Variables:

- $x_{ijk} \in \{0,1\}$: $x_{ijk}=1 \Rightarrow$job j is selected & scheduled on location (i,k)
- $C_{ik} \in \mathbb{R}^+$: compl. time of the job selected & scheduled on loc. (i,k)
- $U^P_{ijk} \in \{0,1\}$: job j is selected & scheduled on loc. (i,k) and is **POTENTIALLY** tardy (to avoid the presence of variables $e_{i,j,k}$ for early jobs)

Exact methods

MIP formulation

Variables:

- $x_{ijk} \in \{0,1\}$: $x_{ijk}=1 \Rightarrow$job j is selected & scheduled on location (i,k)
- $C_{ik} \in \mathbb{R}^+$: compl. time of the job selected & scheduled on loc. (i,k)
- $U^P_{ijk} \in \{0,1\}$: job j is selected & scheduled on loc. (i,k) and is **POTENTIALLY** tardy (to avoid the presence of variables $e_{i,j,k}$ for early jobs)

Objective: min $\sum_{j=1}^{N} \sum_{(i,k)\in E} w_j \, U^P_{ijk} \Rightarrow \sum_{j=1}^{N} \sum_{(i,k)\in E} w_j \, U_{ijk}$

Exact methods

MIP formulation

Variables:

- $x_{ijk} \in \{0,1\}$: $x_{ijk}=1 \Rightarrow$job j is selected & scheduled on location (i,k)
- $C_{ik} \in \mathbb{R}^+$: compl. time of the job selected & scheduled on loc. (i,k)
- $U^P_{ijk} \in \{0,1\}$: job j is selected & scheduled on loc. (i,k) and is **POTENTIALLY** tardy (to avoid the presence of variables $e_{i,j,k}$ for early jobs)

Objective: min $\sum_{j=1}^{N} \sum_{(i,k)\in E} w_j \, U^P_{ijk} \Rightarrow \sum_{j=1}^{N} \sum_{(i,k)\in E} w_j \, U_{ijk}$

Constraints:

- Block structure: **fill from the RIGHT to LEFT**
- SPT order on each machine schedules
- A job j that is not "potentially" tardy is necessarily early and fulfils the **due date constraint**:
- Standards constraints: a job cannot be duplicated, only selected jobs can be tardy...

2 Exact methods

- MIP
- Column generation

Exact methods

Column generation: Master problem

We are extending the approach of Van Den Akker[2].

The master problem: set-covering problem with an exponential number of binary variables and a linear number of constraints.

[2] J. M. Van Den Akker, J. A. Hoogeveen, and S. L. Van de Velde, "Parallel machine scheduling by column generation," *Operations research*, vol. 47, no. 6, pp. 862–872, 1999.

Exact methods

Column generation: Master problem

We are extending the approach of Van Den Akker[2].

The master problem: set-covering problem with an exponential number of binary variables and a linear number of constraints.

Let s be a machine schedule (column) defined as a sequence of jobs that can be assigned to any machine.

The jobs in s must be numbered according to SPT.

[2] J. M. Van Den Akker, J. A. Hoogeveen, and S. L. Van de Velde, "Parallel machine scheduling by column generation," *Operations research*, vol. 47, no. 6, pp. 862–872, 1999.

Exact methods

Column generation: Master problem

For a given sequence s, we can define the following constants :

For a given sequence s, we can define the following constants :

$$a_{j,s} = \begin{cases} 1 \text{ if the job } J_j \text{ is in the schedule } s \\ 0 \text{ otherwise} \end{cases}$$

$$t_{j,s} = \begin{cases} 1 \text{ if the job } J_j \text{ is late in the schedule } s \\ 0 \text{ otherwise} \end{cases}$$

Let $C_j(s)$ be the completion time of the job j in the schedule s

$$C_j(s) = \sum_{k=1}^{j} a_{k,s} \frac{p_k}{V^\gamma} \tag{2}$$

Exact methods

Column generation: Master problem

We want to compute a LB to the $\sum_j w_j U_j$, in the **bilevel scenario**

$\Rightarrow$ integrate the constraint " *when the n jobs are selected, they must be scheduled to minimize* $Lex(\sum_j C_j, \sum_j w_j U_j)$ "

Exact methods

Column generation: Master problem

We want to compute a LB to the $\sum_j w_j U_j$, in the **bilevel scenario**

$\Rightarrow$ integrate the constraint " *when the n jobs are selected, they must be scheduled to minimize* $Lex(\sum_j C_j, \sum_j w_j U_j)$ "

$\Rightarrow$ We minimize the $Lex(\sum_j w_j U_j, \sum_j C_j)$ problem in a single level scenario where we have n jobs to select.

Exact methods

Column generation: Master problem

We want to compute a LB to the $\sum_j w_j U_j$, in the **bilevel scenario**

$\Rightarrow$ integrate the constraint " *when the n jobs are selected, they must be scheduled to minimize* $Lex(\sum_j C_j, \sum_j w_j U_j)$ "

$\Rightarrow$ We minimize the $Lex(\sum_j w_j U_j, \sum_j C_j)$ problem in a single level scenario where we have n jobs to select.

For one schedule s, we have the cost c_s defined as follows:

$$c_s = \sum_{j=1}^{N} w_j t_{j,s} + \alpha \sum_{j=1}^{N} C_j(s) \tag{3}$$

where: $\alpha = \frac{1}{\sum_{j=1}^{N}(w_j+p_j)}$

Exact methods

Column generation: Master problem

We want to compute a LB to the $\sum_j w_j U_j$, in the **bilevel scenario**

$\Rightarrow$ integrate the constraint " *when the n jobs are selected, they must be scheduled to minimize* $Lex(\sum_j C_j, \sum_j w_j U_j)$ "

$\Rightarrow$ We minimize the $Lex(\sum_j w_j U_j, \sum_j C_j)$ problem in a single level scenario where we have n jobs to select.

For one schedule s, we have the cost c_s defined as follows:

$$c_s = \sum_{j=1}^{N} w_j t_{j,s} + \alpha \sum_{j=1}^{N} C_j(s) \tag{3}$$

where: $\alpha = \frac{1}{\sum_{j=1}^{N}(w_j + p_j)}$

Let be $x_s \in \{0, 1\}$ where: $x_s = \begin{cases} 1 \text{ if the column } s \text{ is selected} \\ 0 \text{ otherwise} \end{cases}$

We have the following master's problem formulation :

$$\text{Minimize} \quad \sum_{s^1 \in S_1} c_{s^1} x_{s^1} + \sum_{s^0 \in S_0} c_{s^0} x_{s^0} \tag{4}$$

subject to:

$$\sum_{s^0 \in S_0} a_{j,s^0} x_{s^0} + \sum_{s^1 \in S_1} a_{j,s^1} x_{s^1} \leqslant 1 \quad \forall j = 1, \ldots, N \tag{5a}$$

$$\sum_{s^0 \in S_0} x_{s^0} \leqslant N_0 \tag{5b}$$

$$\sum_{s^1 \in S_1} x_{s^1} \leqslant N_1 \tag{5c}$$

$$\sum_{j=1}^{N} \left(\sum_{s^0 \in S_0} a_{j,s^0} x_{s^0} + \sum_{s^1 \in S_1} a_{j,s^1} x_{s^1} \right) = n \tag{5d}$$

We compute the reduced cost r_s of any machine schedule s as follow:

$$r_s = c_s - \tilde{\lambda} - \sum_{j=1}^{N} \lambda_j \, a_{j,s} - \lambda_{N+1} \sum_{j=1}^{N} a_{j,s} \tag{6}$$

We solve the *pricing problem*, that involves finding the column with the minimum reduced cost.

- We achieve this through dynamic programming in $\mathcal{O}\left(\sum_{j=1}^{N} p_j \times N^2\right)$ time and then by backtracking.
- Using a heuristic that generate column with negative reduced cost.

We identify the more fractional job j.

Let the fractional completion time:

$$\bar{C}(j) = \sum_{s \in S^*(j)} C_j(s) x_s^*$$

Exact methods

Branch and Bounds

We identify the more fractional job j.

Let the fractional completion time:

$$\bar{C}(j) = \sum_{s \in S^*(j)} C_j(s) x_s^*$$

Branching scheme, create 5 nodes :

- j is on high speed machines & $C_j \leqslant \lfloor \bar{C}(j) \rfloor$
- j is on high speed machines & $C_j \geqslant \lfloor \bar{C}(j) \rfloor + 1$
- j is on low speed machines & $C_j \leqslant \lfloor \bar{C}(j) \rfloor$
- j is on low speed machines & $C_j \geqslant \lfloor \bar{C}(j) \rfloor + 1$
- Don't take job j

Exact methods

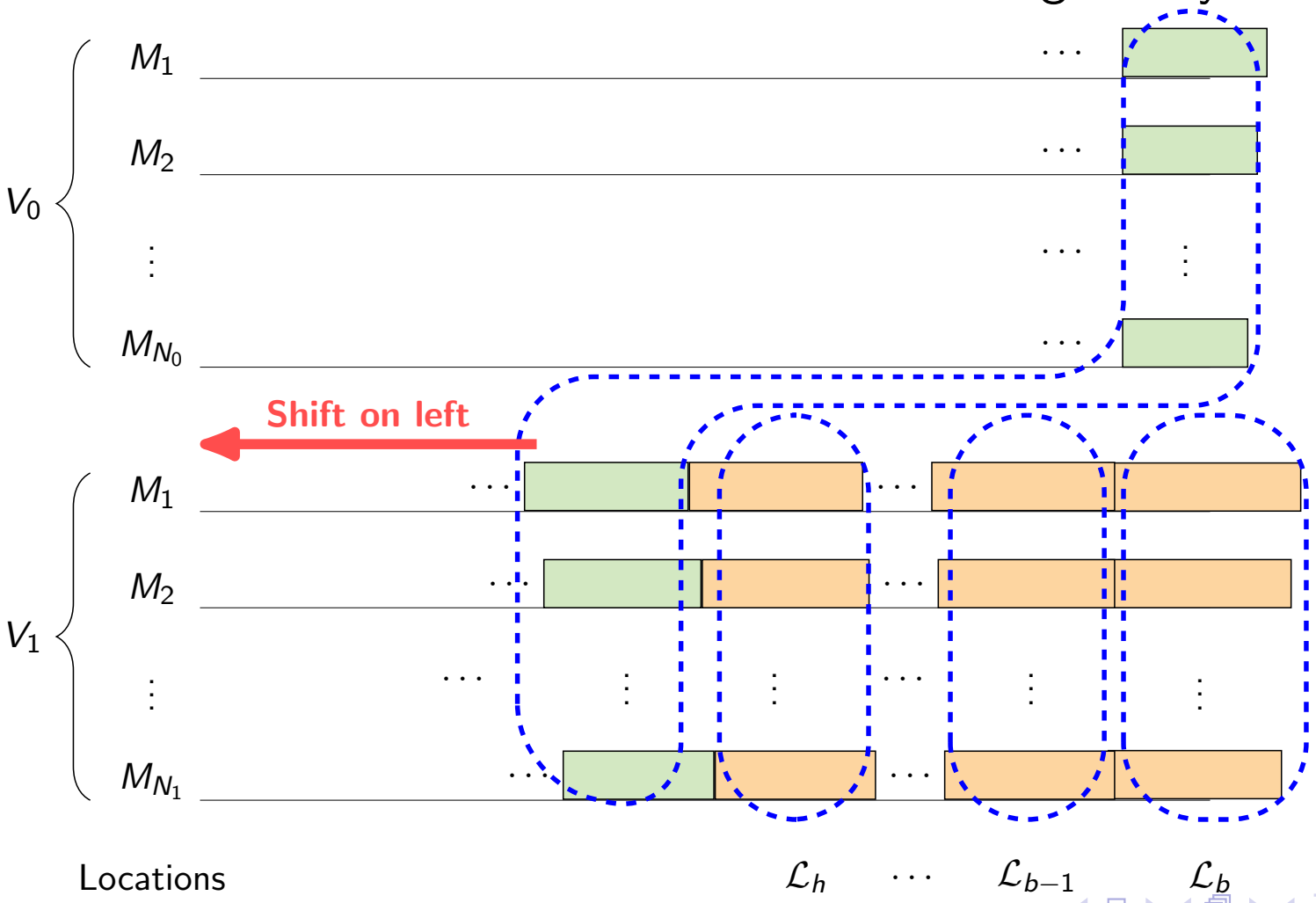
Branch and Bounds

When we reach a node with n selected jobs, we **start solving the bilevel** problem and branch on the **location available** given by **block structure**.

Exact methods

Branch and Bounds

When we reach a node with n selected jobs, we **start solving the bilevel** problem and branch on the **location available** given by **block structure**.



3 Computational results

Computational results

25 types of instances with: $p_j \in [\![1, 100]\!]$, $w_i \in [\![1, 100]\!]$ and $d_i \in [\![p_j \times (1 - tf - \frac{rdd}{2}), p_j \times (1 - tf + \frac{rdd}{2})]\!]$ where $tf, rdd \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$

MIP (125 instances)

	MIP			
N	#OPT	T_{min}	T_{avg}	T_1
15	125	0.007	0.012	0.020
20	125	0.017	0.031	0.154
30	125	0.015	0.023	0.035
40	125	0.143	1.85	20.5
50	122	0.312	41.1	600
60	110	0.616	109	600
70	88	1.68	257	600
80	59	4.88	411	600
90	35	17.5	501	600

Table: Computing time for MIP method with 2 machines and $n = \frac{N}{2}$

	MIP			
N	#OPT	T_{min}	T_{avg}	T_1
15	125	0.009	0.014	0.039
20	125	0.025	0.057	0.254
30	125	0.074	0.279	2.46
40	125	0.126	5.54	120
50	116	0.38	113	600
60	68	1.29	347	615
70	35	2.66	474	600
80	17	16.1	545	600
90	11	32.3	570	600

Table: Computing time for MIP method with 4 machines and $n = \frac{N}{2}$

4 Conclusions

Conclusions

- Some tests are running to measure the performance of the column generation

Conclusions

- Some tests are running to measure the performance of the column generation
- Other heuristic methods are being considered, using machine learning to approximate follower's decision

Conclusions

- Some tests are running to measure the performance of the column generation
- Other heuristic methods are being considered, using machine learning to approximate follower's decision
- Other problems are considered with several followers

Conclusions

Thank you for your time!

Questions?

Block structure:

All locations, except perhaps those in the first block are occupied (**fill from the RIGHT to LEFT**):

$$\sum_{j=1}^{N} \sum_{(i,k)\in E\setminus\mathcal{L}_1} x_{ijk} = |E \setminus \mathcal{L}_1| = |E| - |\mathcal{L}_1| \tag{7}$$

The first block may not be totally filled. There are exactly $|\mathcal{L}_1| + n - |E|$ jobs in the first block:

$$\sum_{j=1}^{N} \sum_{(i,k)\in\mathcal{L}_1} x_{ijk} = |\mathcal{L}_1| + n - |E| \tag{8}$$

Processing times must increase between 2 consecutive blocks:

$$\forall h < b, \forall (i,k) \in \mathcal{L}_h, \forall (i',k') \in \mathcal{L}_{h+1}, \sum_{j=1}^{N} x_{ijk}\, p_j \leqslant \sum_{j=1}^{N} x_{i'jk'}\, p_j \tag{9}$$

Block structure:

All locations, except perhaps those in the first block are occupied (**fill from the RIGHT to LEFT**):

$$\sum_{j=1}^{N} \sum_{(i,k)\in E\setminus\mathcal{L}_1} x_{ijk} = |E \setminus \mathcal{L}_1| = |E| - |\mathcal{L}_1| \tag{7}$$

The first block may not be totally filled. There are exactly $|\mathcal{L}_1| + n - |E|$ jobs in the first block:

$$\sum_{j=1}^{N} \sum_{(i,k)\in\mathcal{L}_1} x_{ijk} = |\mathcal{L}_1| + n - |E| \tag{8}$$

Processing times must increase between 2 consecutive blocks:

$$\forall h < b, \forall (i,k) \in \mathcal{L}_h, \forall (i',k') \in \mathcal{L}_{h+1}, \sum_{j=1}^{N} x_{ijk}\, p_j \leqslant \sum_{j=1}^{N} x_{i'jk'}\, p_j \tag{9}$$

A job j that is not "potentially" tardy is necessarily early and checks the **due date constraint**:

$$\forall (i,k) \in E, C_{ik} \leqslant \sum_{j=1}^{N} x_{ijk}\, d_j + \left(\sum_{j=1}^{N} U_{ijk}^{P}\right) HV \tag{10}$$

Other classical constraints:

Computes all completion times:

$$\forall (i,k) \in E, C_{ik} = C_{i,k-1} + \sum_{j=1}^{N} x_{ijk} \frac{p_j}{V_i} \tag{11}$$

where $\forall i, C_{i,0} = 0$

Only a present job can be tardy:

$$\forall j, \forall (i,k) \in E, U_{ijk}^{P} \leqslant x_{ijk} \tag{12}$$

A job cannot be duplicated:

$$\forall j, \sum_{(i,k)\in E} x_{ijk} \leqslant 1 \tag{13}$$

Each location can have at most one job:

$$\forall (i,k) \in E, \sum_{j=1}^{N} x_{ijk} \leqslant 1 \tag{14}$$