FEOR – Optimisation II

Introduction to Scheduling

Agnès Gorge

Africa Business School

What do we mean by scheduling ?

Scheduling is the allocation of shared resources over time to competing activities

Machine scheduling problem :

- **Activities → jobs = a set of operations**
- **Resources → machines** that can process at most one operations at a time

Objective :

- Min makespan
- Min cost
- Max flowtime
- ….

Inputs

- A set of jobs
- A set of machines
- Each job = a set of operations to be processed on dedicated machines(s) & their associated processing time
- Various constraints (precedence, changeover, …)

Scheduling

Outputs

An allocation of jobs x resources over time

= Assignment of each operation on a resource

+ Sequencing

+ Starting time

A simple example

Jobs

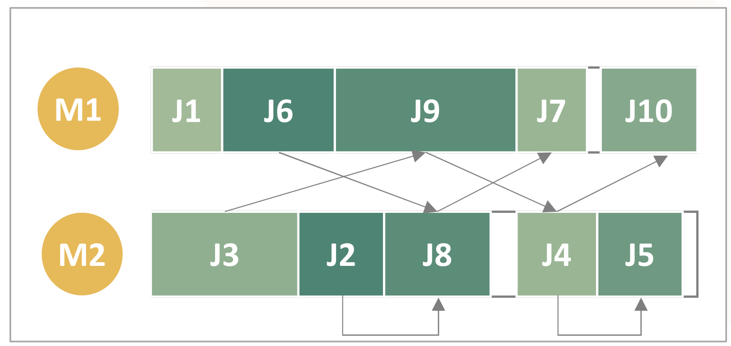
- 10 jobs to allocate either on M1, or on M2
- Each job = one single operation
- Each job can be done on the two machines but with different duration

Machine

M1 M2

Constraints

- Non preemptive
- Precedence constraints



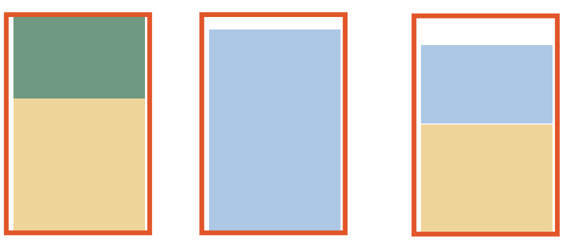
What's the difference between planning & scheduling ?

Planning	Scheduling

Planning

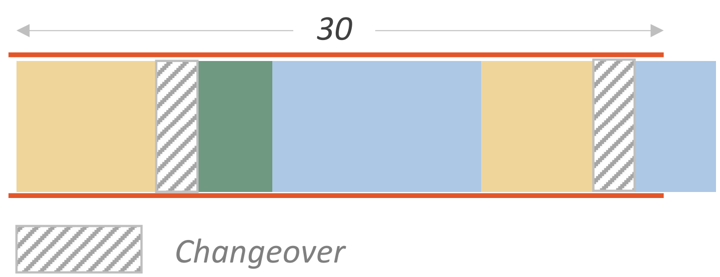
- **Tactical (mid-term)**
- **Discrete time**
- For each time step, look if the capacity is sufficient to complete a series of task
- "Think globally" = the task will be done during the time step but we don't know exactly when
- In the case where the order impacts the capacity to produce (typically: changeover) → risk of being over-optimist (~ constraint relaxation)
- Small time step implies better precision but higher computation time

3 time-steps of 10



Scheduling

- **Operational (short-term)**
- **Continuous time (**No need to discretize)
- Each task is allocated to the resource within a precise time slot
- Make it possible to consider changeover
- 2 key **decision variables** :
  - Loading = assignment to resources
  - Sequencing = order in which they are carried out (notion of sucessor and/or predecessor)



4 majors fields of applications of scheduling problems

1 **Production**

Flexible manufacturing
Assembly problems
With or without transportation

2 **Computer science**

CPU Scheduling
Mono or multi processor systems

3 **Project management**

Multi-resource scheduling
Precedence constraints
PERT

4 **Workforce management**

Timetabling scheduling problems
(education, health, transport, …)

Key characteristics

Operations characteristics

- **Preemption or not**
- **Processing time**

Machine type

- **Identical**
- **Uniform (different yield)**
- **Homogeneous**

Constraints

- **Earliest date or deadline**
- **Changeover or setup**
- **Same machine**
- **....**

Operations per job

- **Single**
- **Ordered**
- **Precedence** (represented through an acyclic graph)
- **Non ordered** = totally free

Work process = Possible machine per operations

- **Only one dedicated machine**
- **Several but with different processing time**
- **Several with same processing time**

Objective

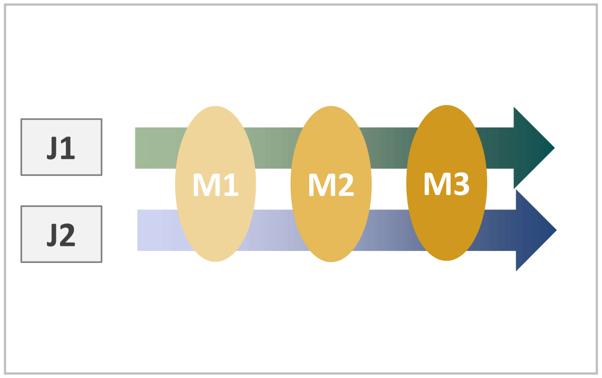
Let C_i be the completion time of job J_i : objective = f(C)

- **Cost function**
- **Makespan**
- **Weighted flowtime = $\Sigma_i w_i C_i$**
- **...**

3 famous scheduling problems

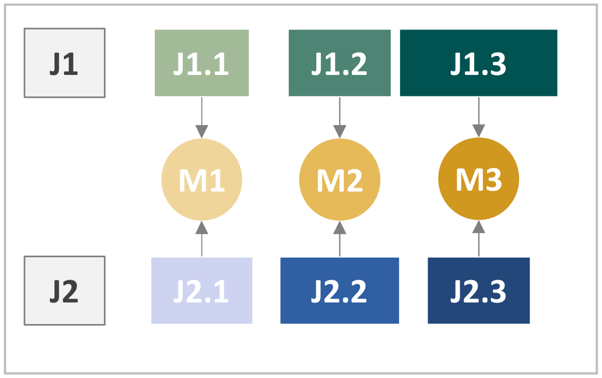
Flow shop

- Each job J_i consists of the same set of operations O_{ij} with various processing time
- Dedicated machines
- Strict order : $O_{i1} \rightarrow O_{i1} \rightarrow \dots \rightarrow O_{in}$



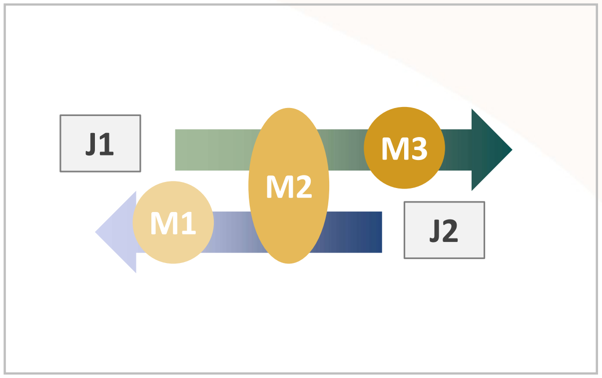
Open shop

- Each job J_i consists of a set of operations O_{ij}
- Dedicated machines
- No precedence relations between operations



Job shop

- Each job J_i consists of a specific set of operations O_{ij}
- Dedicated machines
- Strict order : $O_{i1} \rightarrow O_{i1} \rightarrow \dots \rightarrow O_{in}$



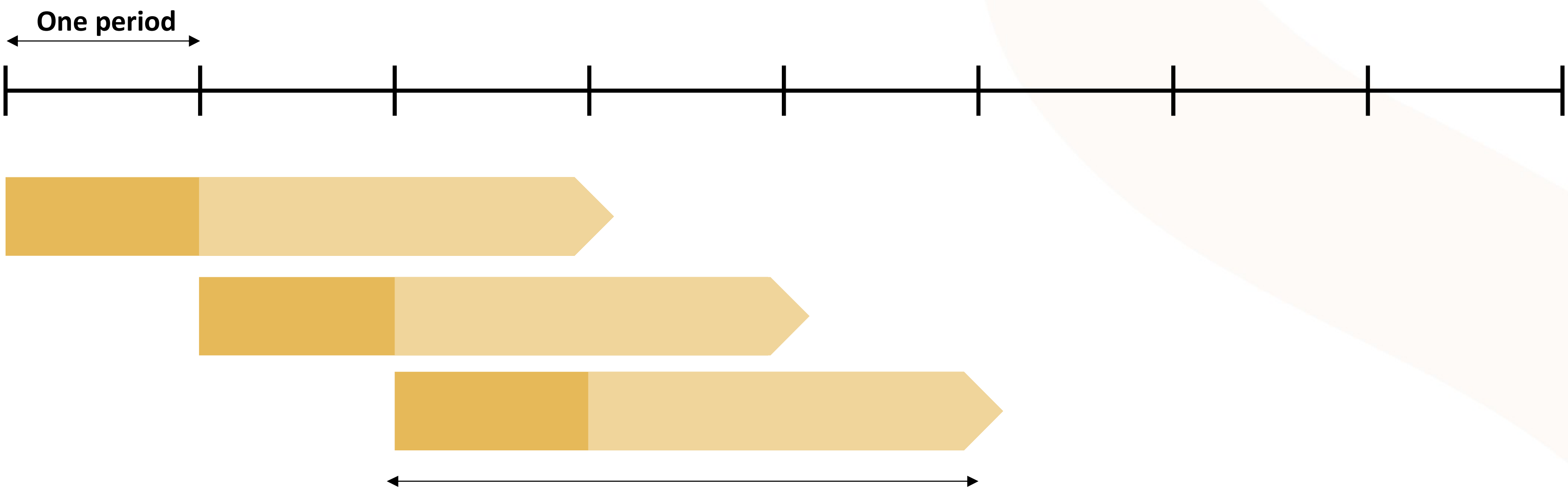
Mass production — Customisation

Various way of (re) scheduling to cope with uncertainty

	The scheduling scheme		The rescheduling scheme	
Trigger	Periodic (off-line)	New task appears (on-line, or real-time)	• Periodic (if period < time-horizon) • Adaptive • Reactive	
Considering uncertainty	• No = Deterministic opt°	• Heuristics (= rules) • Optimal policies → Dyn. programming *Rules & policies may incorporate or not uncertainties*	• No = Deterministic	
Methodology	• Yes = Proactive = Robust or stochastic opt°		Deterministic optimization	Heuristics (= rules)
Objective	• Performance • Robustness • Stability		• Performance • Disturbance	
Decision scope	• All		• All • A part of them (= recourse decisions)	None (= repair). *Ex : match-up*

Example of a periodic off-line (re)scheduling

LW = Look-ahead window



LW trade-off :
- Short → myopic optimization
- Large → performance & data issues

How to solve scheduling problems which are amongst the hardest of NP-hard problems

Exact methods	For specific problems, there exist exact polynomial algorithm	- **Ex: Johnson's algorithm for min makespan on a flow shop with 2 machines** - **Shortest path if an infinite number of machine with predence graph (Central Scheduling Problem)**
	For small instances, you can try a MILP	**3 widely used general formulations :** - **Time-indexed formulation** - **Rank-based formulation** - **Disjunctive formulation**
Approximation methods	Meta-heuristics	- Genetic algorithm - Simulated annealing - Tabu search - ...
	Priority rules *Also work for dynamic problems*	- **FCFS =** First Come First Serve - **SPT =** Shortest Processing Time - **EDD =** Earliest Due Date - **CR =** Critical Ratio (= Processing time / time until due)
	Dedicated algorithm	- **Active schedule generation heuristics** - **Shifting bottleneck for job shop**

A creuser

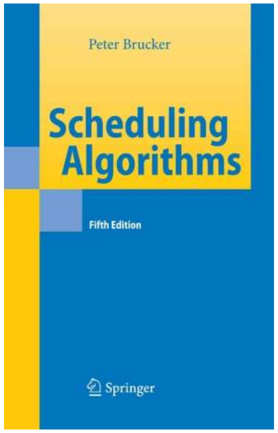
J. Adamas, E. Balas, D. Zawack "The shifting booeneck procedure for job shop scheduling"
Wen-Yang Ku, J. C. Beck : "Mixed Integer Programming Models for Job Shop Scheduling: A Computational Analysis"

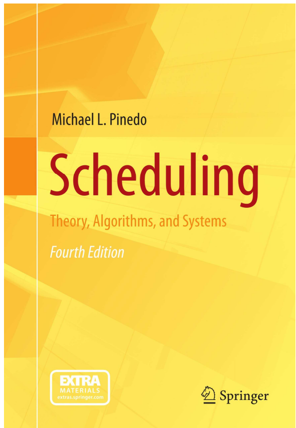
Job-shop example : 3 possible formulations as a MIP

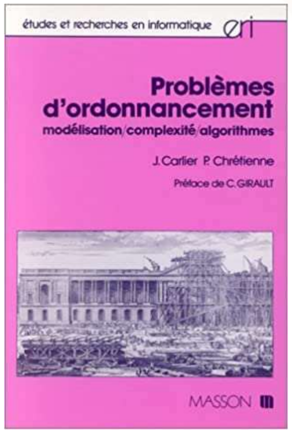
- j=1,…,J jobs of duration *d[j]*
- m=1,…,M machines
- t=1,…,T time step

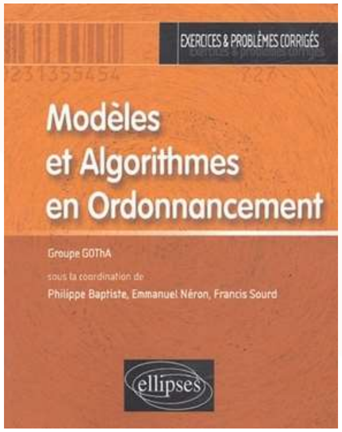
	Variables	Key constraints
Time-indexed	• **X[t][j][m] in {0,1}** *#equal to 1 if job j starts at time t at machine i.*	• **Σ[j][t–d[j] +1 ≤ t' ≤ t] X[t][j][m] ≤ 1** *# for all j, m*
Rank-indexed	• **X[j][m][k] in {0,1}** #equal to 1 if job j is scheduled at the k-th position on machine I • **Y[m][k]** #start time of the job at the k-th position of machine i	• **Y[m][k] + Σ[j]** ***d[j]*** **Y[m][k] X[j][m][k] ≤ Y[m][k+1]** *# for all m, k*
Disjunctive	• **X[j][j'] [m] in {0,1}** *#equal to 1 if j precedes j' on machine m* • **Y[j][m]** *#start time of job j on machine m*	• **Y[j][m] +** ***d[j]*** **≤ Y[j'][m] +** ***M*** **X[j][j'] [m]** *#for all j, j', m*

References


Peter Brucker
Scheduling Algorithms
Fifth Edition
Springer

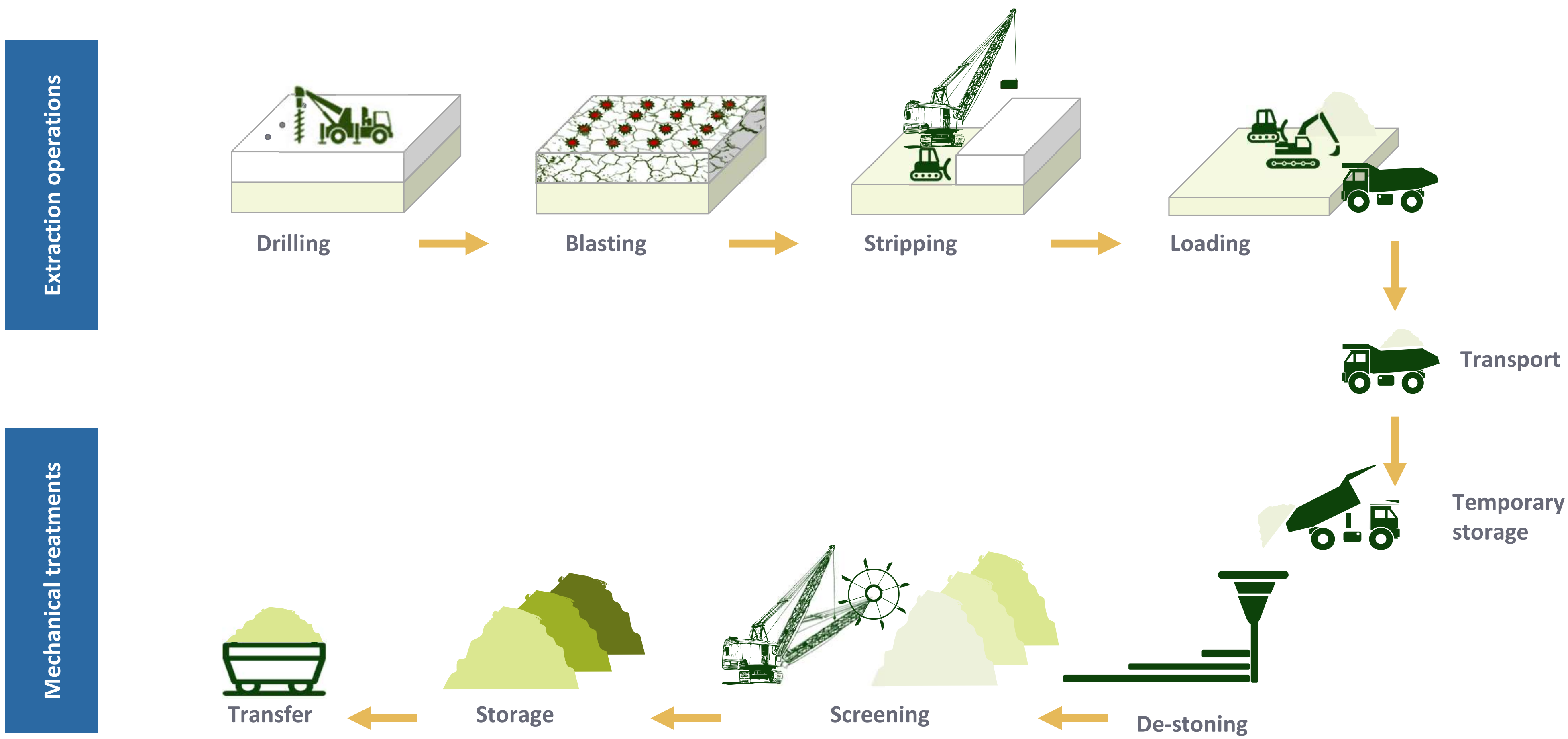

Michael L. Pinedo
Scheduling
Theory, Algorithms, and Systems
Fourth Edition
EXTRA MATERIALS
Springer


études et recherches en informatique
Problèmes d'ordonnancement
modélisation/complexité/algorithmes
J. Carlier P. Chrétienne
Préface de C. GIRAULT
MASSON

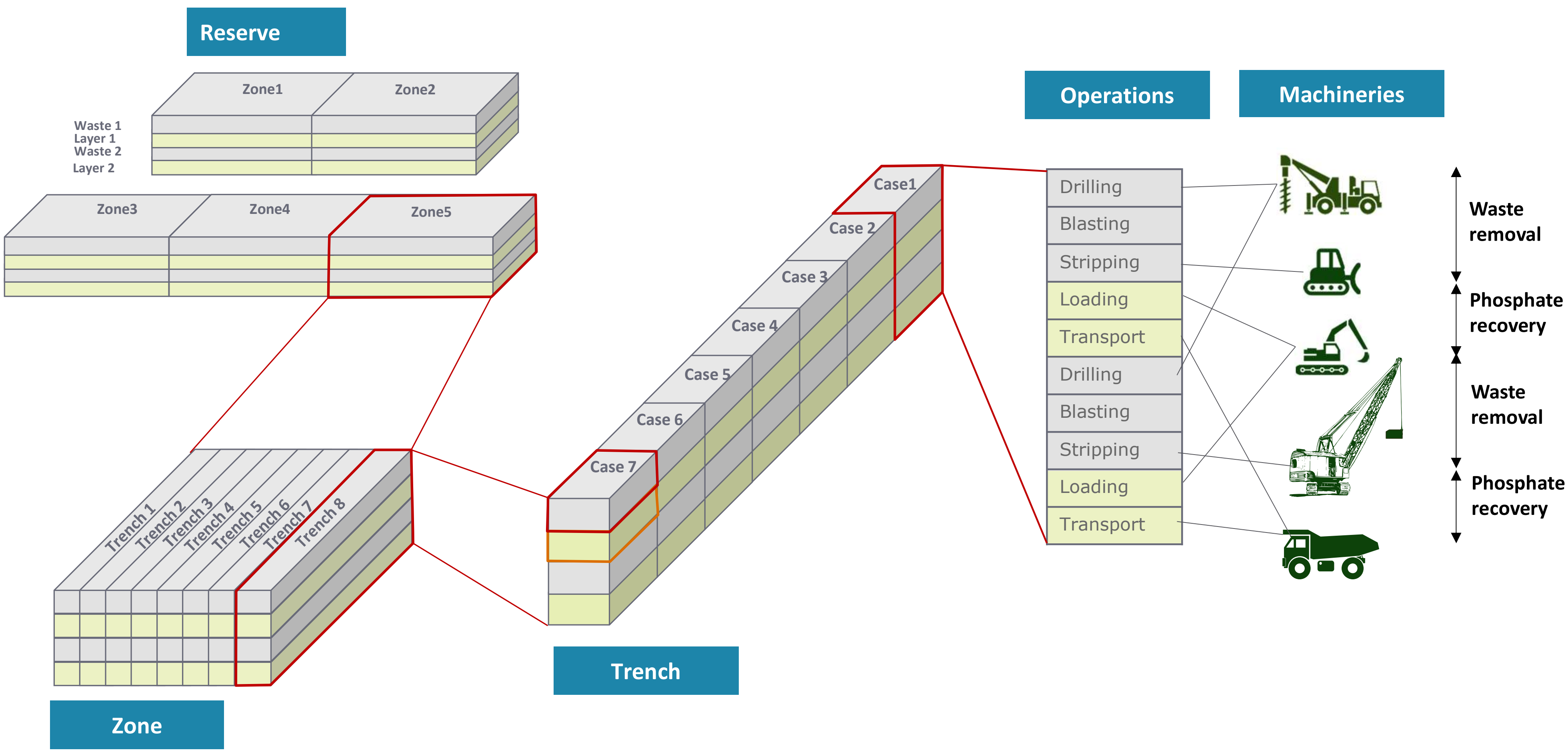

EXERCICES & PROBLÈMES CORRIGÉS
Modèles et Algorithmes en Ordonnancement
Groupe GOThA
sous la coordination de
Philippe Baptiste, Emmanuel Néron, Francis Sourd
ellipses

PROJET : MINE SCHEDULING

Phosphate extraction consists of a sequence of operations

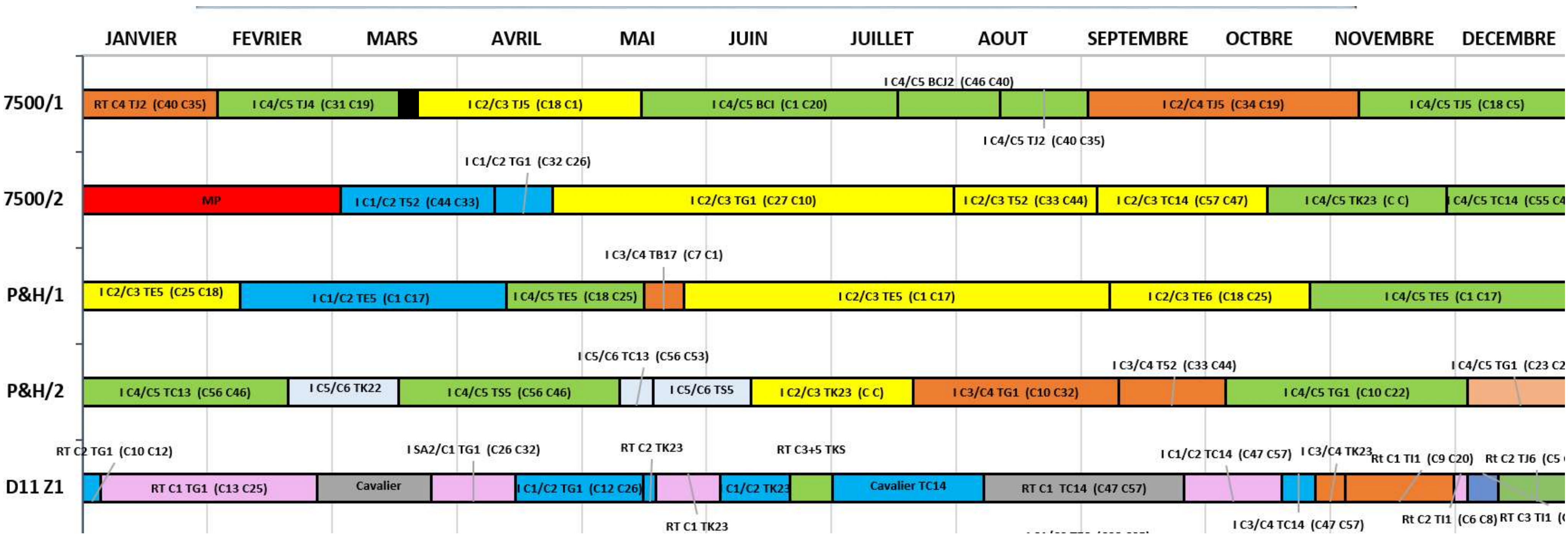


These operations can take place in multiple areas / trenchs / layer

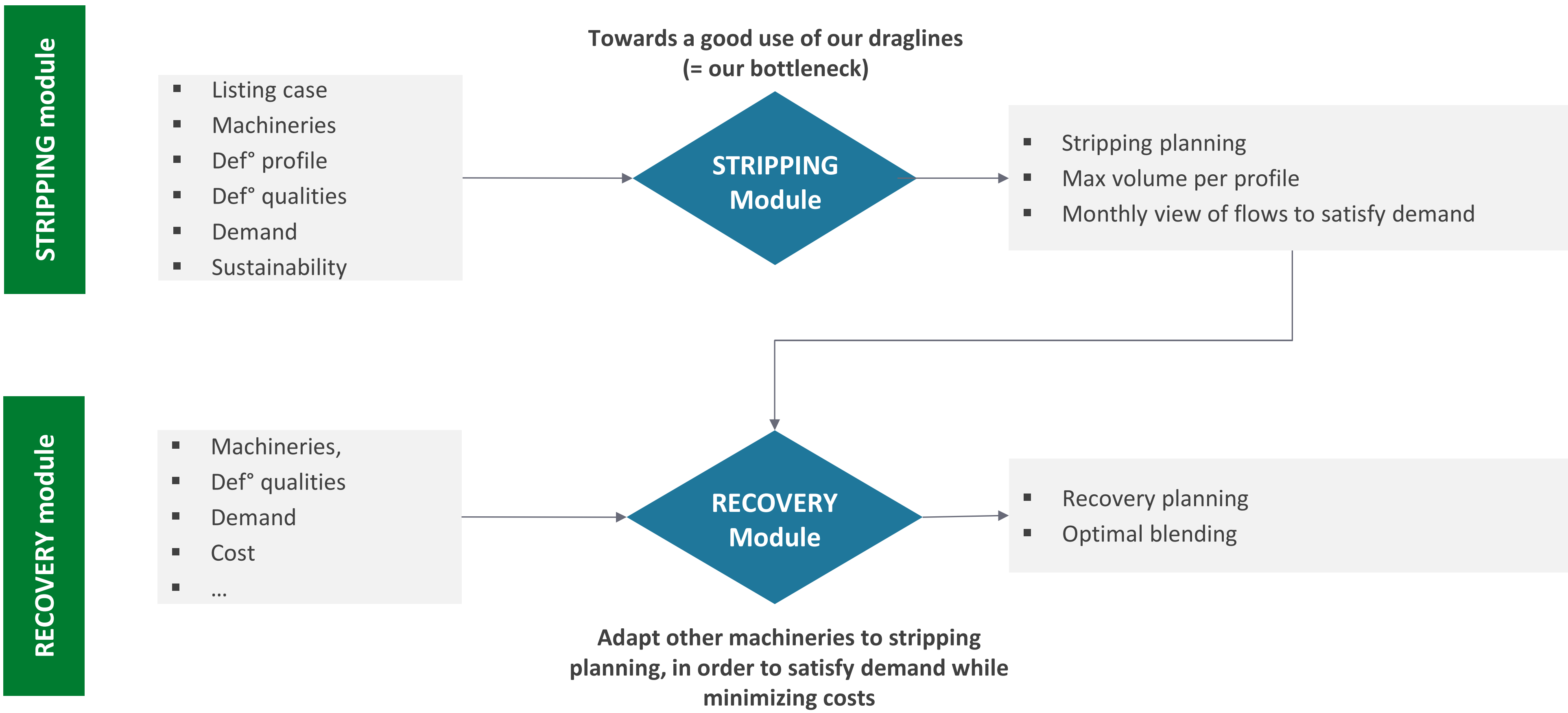


Mine planning involves scheduling the operations of each machine

Illustration



We have broken down the problem in order to make it tractable



Quelle fonction objectif pour le module décapage ?

Min storage recovery for demand satisfaction

+

Supply < Demand

Max revenue

Obj = Max ∑ coef x prod

Push mode

Supply > Demand

Min OPEX
Short-term view

Min TU dragline

We only produce the demand and nothing more

Which idle cost ?

Pull mode

Max use of my resources
Long-term view

Max use of my reserve

Not optimisable on the long term :
what I do not use this year, I will use it next year

Max use of my machineries

Since the « lost time » will not be recovered lately

Hybrid mode

Mine scheduling

Problem description

Our mine is constituted of **a list of job** to perform potentially : j = 1, …, J, corresponding to a geographical zone, on a given trench and layer characterized by a volume volume[j] of waste to move.

A job can be done by multiple machine m, with various duration. Maximum one machine can be used for a job & maximum one job per machine at the same time.

Each job provides a production (of phosphate for instance).

Each machine need time to move from one job to another : switchingTime[m][j][j']

Some job can start only when another job (called Predecessor) has been finished.

Objectif : maximize production on the given horizon

Simplification to consider

- **For objective :** consider only the production of the task that are totally finished at the end of the horizon
- **Bonus :** Consider even the « not-finished » task, with a prorata rule (ex : if 20% of the duration has been spent, then 20% of the production is available)
- **In reality :** the production follows a demand constraint (per rock type and time-period) and the objective function is the cost
- **Bonus :** What is the risk/limits of maximizing the production, or minimizing the cost ?

What we expect from you

1. **Present and explain the formulation (you don’t have to stick to the proposed formulation, do not hesitate if you want to innovate)**
   - What are decision variables Vs. Simulation variables Vs. State variables ?
   - You may suggest ways to reduce the size of the problem
2. **Present your results on the small Data Set (production, computation time) when the time horizon varies**
3. **Present the limits of this simplified way of modeling the problem**
4. **As a conclusion, present 3 key learnings of this module**

Defense :

- 15 min presentation
- 5 min Q&A

Open-office next week to explain the formulation (ex : Friday 6pm)

Formulation

Variables

Per job :

- isOver[j] in {0,1}
- start[j] in [0, horizonDuration]

Per possible pair [j, m] :

- allocation[j][m] in {0,1}
- isFirst[j][m] in {0,1}

Per possible succession : j1 → j2 on machine m

- isSuccessorVar[m][j1][j2] in {0,1}

Objective

Sum[j] production[j] * isOverVar[j]

Constraints

1. Max one allocation per job
2. Max one "isFirst" job per machine
3. Constraint stating than start[j] + duration[m][j] + switchingTime[m][j][j'] ≤ start[j'] if j' is j's successor on m
4. Constraint stating the "isOver" variable
5. Constraint stating that to be allocated, a job shall either be the first, or be the successor of another job
6. Max one successor per job
7. Constraint stating that if isOver = 1, then at least one allocation = 1
8. Constraint stating that IsSuccessor[m][j1][j2] = 1 --> allocation[j1][m] = 1
9. Constraint stating that if start > 0, then then at least one allocation = 1
10. Constraint stating that if Sum allocation = 1, then start >= 1.0 : just to check easily which job has started or not
11. Constraint imposing predecessors : if vPred is the predecessor of j
    1. isOver[j] = 1 → isOver[vPred] = 1
    2. j shall start only once vPred has finished

Data set

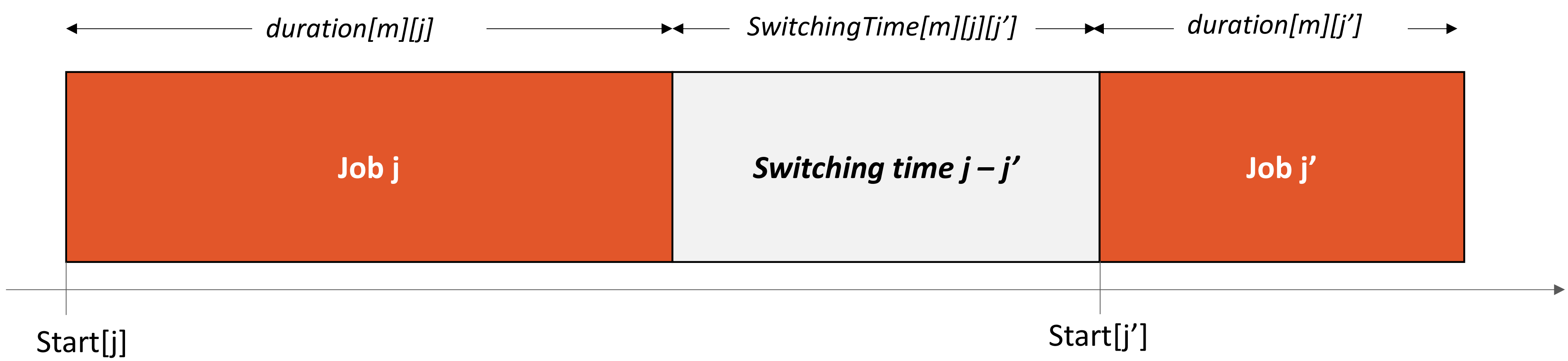
Id	Zone	Trench	Layer	PredecessorId	Volume	Production
Job1	Zone1	1	1		172	58
Job2	Zone1	1	2	Job1	212	43
Job3	Zone1	1	3	Job2	270	82

Job	Zone	Trench	Layer	Machine	Duration
Job1	Zone1	1	1	Bull	17,2
Job2	Zone1	1	2	Bull	28,3
Job3	Zone1	1	3	Bull	54,0
Job3	Zone1	1	3	SmallDragline	45,0

Duration (day)	Machine
30	Bull
	Dragline

Machine	Job1	Job2	SwitchingTime (day)
Bull	Job1	Job2	2
Bull	Job1	Job3	3
Bull	Job1	Job7	1

Constraint stating than start[j] + duration[j] ≤ start[j’] if j’ is j’s successor



x ≤ V + M (1 – b) → start[j] + duration[m][j] + switchingTime[m][j][j’] <= start[j’] + M (1 – isSuccessor[m][j][j’])

Allocation variable

	M1	M2	M3	M4	
Job1					
Job2					
Job3					
I4	1				
I5					
I6	1				
I7					
I8					
I9					
I10	1				

isOver[job] in {0,1} = 1 si job est terminé à la fin de l'horizon

Allocation[Job][m] in {0,1}

Start[job] in R : date de début de mon job

isSuccessor[m][j][j'] in {0,1}

isFirst[m][job] in {0,1} = 1 si job is the first on m

Obj : max Sum[job] *Production[job]* * isOver[job]

Sum[m] Allocation[Job][m] <= 1, for all job

Sum[job] isFirst[m][job] = 1, for all m

Allocation variable

	M1	M2	M3	M4	
Job1					
I2					
I3					
I4	1				
I5					
I6	1				
I7					
I8					
I9					
I10	1				

Obj = max Sum[job] production[job] * isOver[j]

Sum[machines m] Allocation[Job][m] <= 1

Start[job] : date de début de mon job

isFirst[m][job] in {0, 1} = 1 is job is the first on machine m

isSuccessor[m][j][j'] in {0,1} = 1 si machine m passe de j à j'