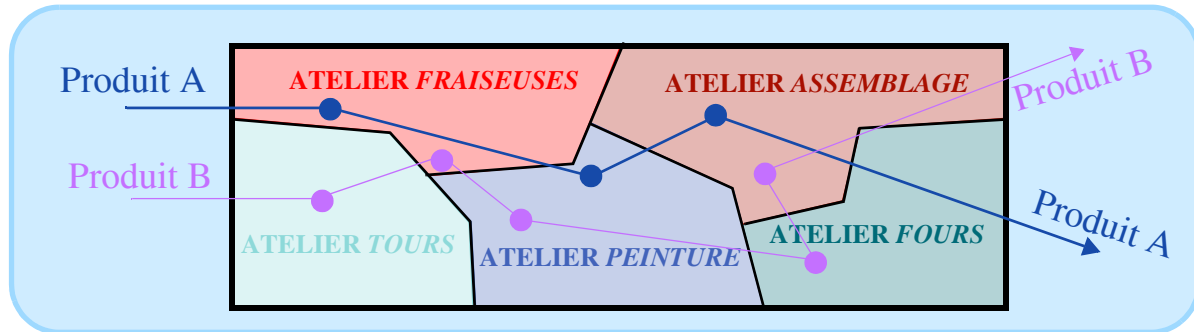


ORDONNANCEMENT EN ATELIERS SPÉCIALISÉS

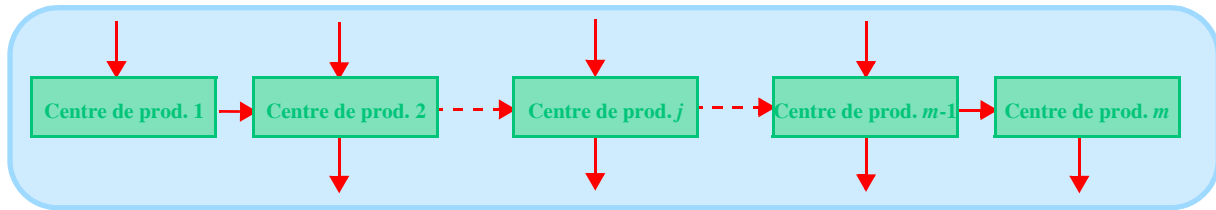
- **Ordonnancement** = *détermination conjointe des dates d'exécution d'un ensemble d'opérations et des ressources mobilisées dans cette exécution*
- **Pb général** dans chaîne logistique (LP, appro/prod synchrone, industries de process, projet)
- **Solutions** \Rightarrow performance/survie
- **Contexte**:
 - \mathcal{E} commandes ou OF pour produire $q_i \geq 1$ de la référence i / CT, découpage temporel fin
 - Système Productif en **AS**, *job shop*, **atelier à cheminements multiples**)



- pb NP dur \Rightarrow utilisation d'heuristiques & qq algo de solution exacte (pb simples)

• **Terminologie**

- **Tâches** (= job = OF \neq projet) \subset \mathcal{E} **opérations** ordre lié par gamme
- **CP** (machine, atelier...); une opération est réalisée par un CP
- préemption = possibilité d'interrompre une opération pour en passer une autre avant de la reprendre plus tard (\Rightarrow problèmes préemptifs)
- Cas particulier des **ateliers à cheminement unique** (*flow shop*) où possibilité $t_{ij} = 0$



Exemple de pb de FS et de solution

OF	1	2	3	4	5	6	7	8	9	10
Machine A	10	12	10	8	0	11	7	6	8	14
Machine B	9	14	17	10	0	12	14	13	0	0
Machine C	13	11	13	12	13	8	14	15	11	13
Machine D	14	17	14	14	15	12	0	8	17	11
Machine E	22	8	13	15	10	19	0	17	11	14



- **Atelier à cheminements libres** (*open shop*): ordre quelconque

- **Cadre d'analyse**

		Problème	
		Statique	Dynamique
Univers	Certain		
	Aléatoire		

SECTION I INTRODUCTION AUX MODÈLES STATIQUES D'ORDONNANCEMENT

I-1 Modèles statiques – Cas des coûts de lancement indépendants de l'ordonnancement retenu

I-1.1 Ordonnancement de n tâches nécessitant l'intervention d'un seul centre de production

I-1.2 Ordonnancement de n tâches nécessitant l'intervention de 2 centres de production

I-1.3 Ordonnancement de 2 tâches nécessitant l'intervention de m centres de production

I-1.4 Ordonnancement de n tâches nécessitant l'intervention de m centres de production

I-1.5 Ordonnancement de n tâches nécessitant l'intervention de m centres de production (cheminement libre – *open shop*)

I-1.6 Ordonnancement de n tâches nécessitant l'intervention de m centres de production (ordre de passage quelconque)

I-2 Modèles statiques: cas du coût de lancement total variable avec l'ordonnancement retenu

I-3 Tentative de caractérisation de l'approche statique

SECTION II L'APPROCHE ALÉATOIRE DYNAMIQUE

SECTION I INTRODUCTION AUX MODÈLES STATIQUES D'ORDONNANCEMENT

I-1 Modèles statiques – Cas des coûts de lancement indépendants de l'ordonnancement retenu

- Hypothèses communes implicites: ordre intangible, temps de transport et de lancement nuls, temps opératoires certains, pas de recouvrement

I-1.1 Ordonnancement de n tâches nécessitant l'intervention d'un seul centre de production

- Durée d'exécution des tâches indépendant de l'ordonnancement
- critères d'évaluation \Rightarrow ordo \neq
- Pb intéressant: goulot d'étranglement

I-1.1.1 L'ordonnement suivant la règle du temps opératoire minimum (règle TOM)

I-1.1.1.1 Exemple introductif

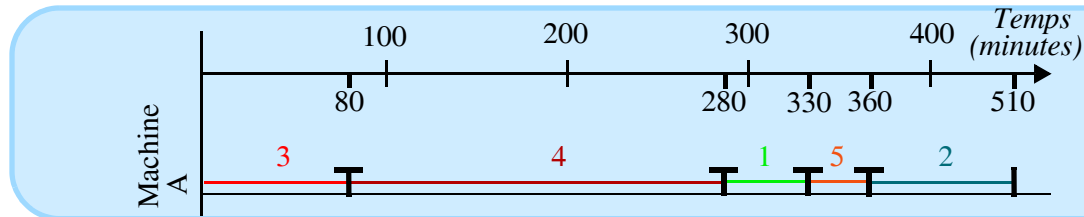
- Exemple.

Tâche i	1	2	3	4	5
Temps opératoire t_i (en centième d'heure)	50	150	80	200	30

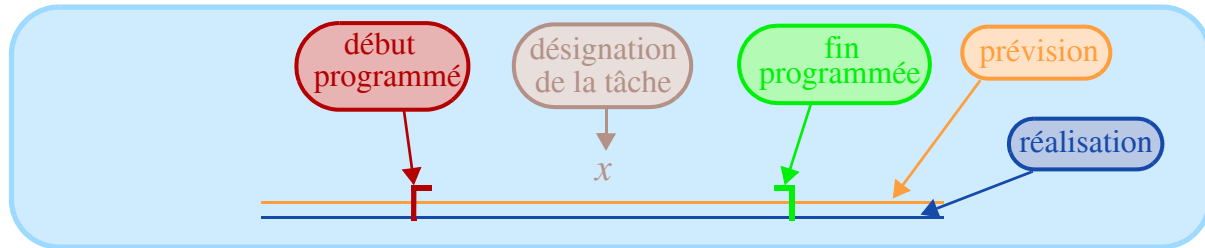
- Solution possible:

Ordre de passage j	1	2	3	4	5
Tâche programmée j	3	4	1	5	2
Temps d'exécution T_j	80	200	50	30	150
Date A_j de fin de la tâche j	80	280	330	360	510

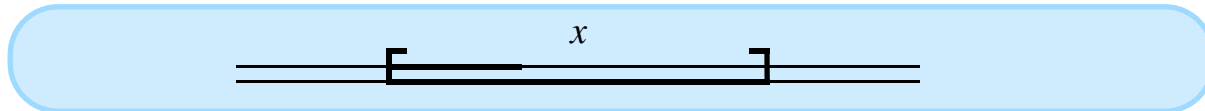
I-1.1.1.2 Graphique de Gantt



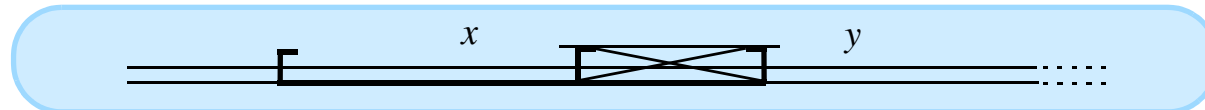
- **Graphique de Gantt** = Diagramme de Gantt = technique de visualisation utilisation moyens productifs et/ou de l'avancement de l'exécution de tâches (Gantt, 1917; prêtres égyptiens)
- **Conventions**



- **Réalisation**
 - **Dépassement de quantités** produites



- **Dépassement de temps**



- **Conventions**: Z (aucun travail exécuté), A (exécutant absent), M (manque de matière première), R (réparation).

I-1.1.1.3 La règle TOM

- Dans exemple: 5! **ordonnements possibles**

- **Date d'achèvement** $A_j = \sum_{h=1}^j T_h$

- **Date moyenne d'achèvement** \bar{A}

Tâche i	1	2	3	4	5
Temps opératoire t_i (en centième d'heure)	50	150	80	200	30

Ordre de passage j	1	2	3	4	5
Tâche programmée j	3	4	1	5	2
Temps d'exécution T_j	80	200	50	30	150
Date A_j de fin de la tâche j	80	280	330	360	510

$$\bar{A} = 312$$

- $\bar{A} = \frac{1}{5} \cdot \sum_{J=1}^5 A_J = \frac{80 + 280 + 330 + 360 + 510}{5} = 312$

Ordonnement en ateliers spécialisés

- Généralisation
$$\bar{A} = \frac{1}{n} \cdot \sum_{j=1}^n A_j = \frac{1}{n} \cdot \sum_{j=1}^n \left(\sum_{k=1}^j T_k \right) = \frac{1}{n} \cdot \sum_{j=1}^n (n-j+1)T_j$$

Ordonnement en ateliers spécialisés

- Règle d'ordonnement **TOM** (*SPT rule, SOT rule*) minimise \bar{A}

$$T_1 \leq T_2 \leq \dots \leq T_j \leq T_{j+1} \leq \dots \leq T_n$$

- Application

Tâche i	1	2	3	4	5
Temps opératoire t_i (en centième d'heure)	50	150	80	200	30

Ordre de passage j	1	2	3	4	5
Tâche programmée	5				
T_j	30				
A_j	30				

- Règle d'ordonnement **TOM** (*SPT rule, SOT rule*) minimise \bar{A}

$$T_1 \leq T_2 \leq \dots \leq T_j \leq T_{j+1} \leq \dots \leq T_n$$

- Application

Tâche i	1	2	3	4	5
Temps opératoire t_i (en centième d'heure)	50	150	80	200	30

Ordre de passage j	1	2			
Tâche programmée	5	1			
T_j	30	50			
A_j	30	80			

Ordonnement en ateliers spécialisés

- Règle d'ordonnement **TOM** (*SPT rule, SOT rule*) minimise \bar{A}

$$T_1 \leq T_2 \leq \dots \leq T_j \leq T_{j+1} \leq \dots \leq T_n$$

- Application

Tâche i	1	2	3	4	5
Temps opératoire t_i (en centième d'heure)	50	150	80	200	30

Ordre de passage j	1	2	3		
Tâche programmée	5	1	3		
T_j	30	50	80		
A_j	30	80	160		

Ordonnement en ateliers spécialisés

- Règle d'ordonnement **TOM** (*SPT rule*, *SOT rule*) minimise \bar{A}

$$T_1 \leq T_2 \leq \dots \leq T_j \leq T_{j+1} \leq \dots \leq T_n$$

- Application**

Tâche i	1	2	3	4	5
Temps opératoire t_i (en centième d'heure)	50	150	80	200	30

Ordre de passage j	1	2	3	4	
Tâche programmée	5	1	3	2	
T_j	30	50	80	150	
A_j	30	80	160	310	

Ordonnement en ateliers spécialisés

- Règle d'ordonnement **TOM** (*SPT rule, SOT rule*) minimise \bar{A}

$$T_1 \leq T_2 \leq \dots \leq T_j \leq T_{j+1} \leq \dots \leq T_n$$

- Application

Tâche i	1	2	3	4	5
Temps opératoire t_i (en centième d'heure)	50	150	80	200	30

Ordre de passage j	1	2	3	4	5
Tâche programmée	5	1	3	2	4
T_j	30	50	80	150	200
A_j	30	80	160	310	510

$$\bar{A} = 218$$

- Règle d'ordonnancement **TOM** (*SPT rule, SOT rule*) minimise \bar{A}

$$T_1 \leq T_2 \leq \dots \leq T_j \leq T_{j+1} \leq \dots \leq T_n$$

- Application

Tâche i	1	2	3	4	5
Temps opératoire t_i (en centième d'heure)	50	150	80	200	30

Ordre de passage j	1	2	3	4	5
Tâche programmée	5	1	3	2	4
T_j	30	50	80	150	200
A_j	30	80	160	310	510

$$\bar{A} = 218$$

- Remarques:

- **priorité** varie en sens inverse de valeur du **critère** (c'est général)
- TOM \nrightarrow $V(A)$ mini (ici 174,06 contre 139,05 ordonnancement initial)
- TOM minimise **retard algébrique moyen**: **retard algébrique** ($T_j - d_j$) \neq **retard vrai** $\text{Max}(0, T_j - d_j)$
- **attente** d'une tâche se définit comme l'intervalle de temps séparant l'arrivée d'une tâche dans le système, du début de son exécution
- Si arrivées dynamiques et préemption: TOM minimise \bar{A}

I-1.1.2 La règle TOM pondéré

- Importance \neq (marge financière...)
- Pondération u_i ($u_i \geq 1$) traduisant priorité accordée à i

- **Temps d'attente moyen pondéré** $\bar{A} = \frac{1}{n} \cdot \sum_{j=1}^n u_j A_j$ minimisé par **règle TOM pondéré**

(règle de Smith) $\frac{T_1}{u_1} \leq \frac{T_2}{u_2} \leq \dots \leq \frac{T_j}{u_j} \leq \frac{T_{j+1}}{u_{j+1}} \leq \dots \leq \frac{T_n}{u_n}$

• Exemple

Tâche i	1	2	3	4	5
Temps opératoire t_i	50	150	80	200	30
Pondération u_i	1	2	1	2	3
t_i/u_i	50	75	80	100	10
Ordre de passage de la tâche i	2	3	4	5	1
Ordre de passage j					
Tâche programmée					
T_h/u_h					
$T_h \cdot u_h$					
$\sum_{h=1}^j T_h \cdot u_h$					

• Exemple

Tâche i	1	2	3	4	5
Temps opératoire t_i	50	150	80	200	30
Pondération u_i	1	2	1	2	3
t_i/u_i	50	75	80	100	10
Ordre de passage de la tâche i					1
Ordre de passage j	1				
Tâche programmée	5				
T_h/u_h	10				
$T_h \cdot u_h$	90				
$\sum_{h=1}^j T_h \cdot u_h$	90				

• Exemple

Tâche i	1	2	3	4	5
Temps opératoire t_i	50	150	80	200	30
Pondération u_i	1	2	1	2	3
t_i/u_i	50	75	80	100	10
Ordre de passage de la tâche i	2				1
Ordre de passage j	1	2			
Tâche programmée	5	1			
T_h/u_h	10	50			
$T_h \cdot u_h$	90	50			
$\sum_{h=1}^j T_h \cdot u_h$	90	140			

• Exemple

Tâche i	1	2	3	4	5
Temps opératoire t_i	50	150	80	200	30
Pondération u_i	1	2	1	2	3
t_i/u_i	50	75	80	100	10
Ordre de passage de la tâche i	2	3			1
Ordre de passage j	1	2	3		
Tâche programmée	5	1	2		
T_h/u_h	10	50	75		
$T_h \cdot u_h$	90	50	300		
$\sum_{h=1}^j T_h \cdot u_h$	90	140	440		

• Exemple

Tâche i	1	2	3	4	5
Temps opératoire t_i	50	150	80	200	30
Pondération u_i	1	2	1	2	3
t_i/u_i	50	75	80	100	10
Ordre de passage de la tâche i	2	3	4	5	1
Ordre de passage j	1	2	3	4	
Tâche programmée	5	1	2	3	
T_h/u_h	10	50	75	80	
$T_h \cdot u_h$	90	50	300	80	
$\sum_{h=1}^j T_h \cdot u_h$	90	140	440	520	

• Exemple

Tâche i	1	2	3	4	5
Temps opératoire t_i	50	150	80	200	30
Pondération u_i	1	2	1	2	3
t_i/u_i	50	75	80	100	10
Ordre de passage de la tâche i	2	3	4	5	1
Ordre de passage j	1	2	3	4	5
Tâche programmée	5	1	2	3	4
T_h/u_h	10	50	75	80	100
$T_h \cdot u_h$	90	50	300	80	400
$\sum_{h=1}^j T_h \cdot u_h$	90	140	440	520	920

$$\bar{A} = 422$$

I-1.1.3 Ordonnancement suivant la règle de la date de livraison minimale

- Introduction de **dates de livraison**.

Tâche i	1	2	3	4	5
Date de livraison d_i souhaitée (en centième d'heures)	100	300	410	400	200
Temps opératoire t_i (en centième d'heures)	50	150	80	200	30
Marge $d_i - t_i$	50	150	330	200	170

- **Conséquences** de **TOM** sur retards vrais

Ordre de passage j (règle TOM)	1	2	3	4	5
Tâche programmée	5	1	3	2	4
A_j	30	80	160	310	510
Date de livraison d_j souhaitée	200	100	410	300	400
Retard vrai: $\max(0, A_j - d_j)$	0	0	0	10	110

Retard minimal: 0
 Retard maximal: 110
 Retard moyen: 24

- Minimisation du retard vrai maximum est minimisé par **règle de Jackson** ordonnant par dates \uparrow de livraison $d_1 \leq d_2 \leq \dots \leq d_j \leq d_{j+1} \leq \dots \leq d_n$

• **Application.**

Tâche i	1	2	3	4	5
Date de livraison d_i souhaitée (en centième d'heures)	100	300	410	400	200
Temps opératoire t_i (en centième d'heures)	50	150	80	200	30

Ordre de passage j	1				
Date de livraison d_j souhaitée	100				
Tâche programmée	1				
Temps opératoire T_j	50				
A_j	50				
Retard vrai maximal	0				

- **Application.**

Tâche i	1	2	3	4	5
Date de livraison d_i souhaitée (en centième d'heures)	100	300	410	400	200
Temps opératoire t_i (en centième d'heures)	50	150	80	200	30

Ordre de passage j	1	2			
Date de livraison d_j souhaitée	100	200			
Tâche programmée	1	5			
Temps opératoire T_j	50	30			
A_j	50	80			
Retard vrai maximal	0	0			

- **Application.**

Tâche i	1	2	3	4	5
Date de livraison d_i souhaitée (en centième d'heures)	100	300	410	400	200
Temps opératoire t_i (en centième d'heures)	50	150	80	200	30

Ordre de passage j	1	2	3		
Date de livraison d_j souhaitée	100	200	300		
Tâche programmée	1	5	2		
Temps opératoire T_j	50	30	150		
A_j	50	80	230		
Retard vrai maximal	0	0	0		

- **Application.**

Tâche i	1	2	3	4	5
Date de livraison d_i souhaitée (en centième d'heures)	100	300	410	400	200
Temps opératoire t_i (en centième d'heures)	50	150	80	200	30

Ordre de passage j	1	2	3	4	
Date de livraison d_j souhaitée	100	200	300	400	
Tâche programmée	1	5	2	4	
Temps opératoire T_j	50	30	150	200	
A_j	50	80	230	430	
Retard vrai maximal	0	0	0	30	

• **Application.**

Tâche i	1	2	3	4	5
Date de livraison d_i souhaitée (en centième d'heures)	100	300	410	400	200
Temps opératoire t_i (en centième d'heures)	50	150	80	200	30

Ordre de passage j	1	2	3	4	5
Date de livraison d_j souhaitée	100	200	300	400	410
Tâche programmée	1	5	2	4	3
Temps opératoire T_j	50	30	150	200	80
A_j	50	80	230	430	510
Retard vrai maximal	0	0	0	30	100

Retard minimal: 0
 Retard maximal: 100
 Retard moyen: 26
 $\bar{A} = 260$
 $\sigma = 183,74$

- Remarque: règle de Jackson **minimise** retard max mais **pas le retard moyen** (ici plus faible avec TOM); pas de règle simple pour y parvenir (ici 1 - 5 - 2 - 3 - 4)
- Si arrivée dynamique et préemption: même propriété

I-1.1.4 Ordonnement suivant la règle de la marge minimale

- Ordonnement par valeurs croissantes de marges ($d_i - t_i$) \Rightarrow maximise le retard le plus faible possible $d_1 - T_1 \leq d_2 - T_2 \leq \dots \leq d_j - T_j \leq d_{j+1} - T_{j+1} \leq \dots \leq d_n - T_n$

Ordre de passage j	1	2	3	4	5
$d_j - T_j$	50	150	170	200	330
Tâche programmée	1	2	5	4	3
Temps d'exécution T_j	50	150	30	200	80
A_j	50	200	230	430	510
d_j	100	300	200	400	410
Retard vrai maximal	0	0	30	30	100

Retard minimal: 0

Retard maximal: 100

Retard moyen: 32

$\bar{A} = 284$

$\sigma = 165,6$




I-1.1.5 Modélisation générale par la PLN

I-1.2 Ordonnement de n tâches nécessitant l'intervention de 2 centres de production

- Seul critère repris: *minimisation du temps total d'exécution de tous les travaux*

I-1.2.1 Cas du même ordre de passage sur les centres de production A et B

- Pb de **flow shop** à 2 centres de production
- **Exemple**

Numéro de la tâche i	1	2	3	4	5	
t_{iA}	50	150	80	200	30	
t_{iB}	60	50	150	70	200	
Rang						


- **Algorithme de Johnson**
 - **Étape 1p**: Chercher i dont t_{ij} (avec $j = A$ ou B) est minimum
 - **Étape 2p**:
 - Si $j = A$ placer i à la première place disponible
 - Si $j = B$ placer i à la dernière place disponible
 - **Étape 3p**: Supprimer i des tâches restant à programmer

I-1.2 Ordonnancement de n tâches nécessitant l'intervention de 2 centres de production

- Seul critère repris: *minimisation du temps total d'exécution de tous les travaux*

I-1.2.1 Cas du même ordre de passage sur les centres de production A et B

- Pb de **flow shop** à 2 centres de production
- **Exemple**

Numéro de la tâche i	1	2	3	4	5	
t_{iA}	50	150	80	200	30	
t_{iB}	60	50	150	70	200	
Rang						


- **Algorithme de Johnson**
 - **Étape 1p:** Chercher i dont t_{ij} (avec $j = A$ ou B) est minimum
 - **Étape 2p:**
 - Si $j = A$ placer i à la première place disponible
 - Si $j = B$ placer i à la dernière place disponible
 - **Étape 3p:** Supprimer i des tâches restant à programmer

I-1.2 Ordonnement de n tâches nécessitant l'intervention de 2 centres de production

- Seul critère repris: *minimisation du temps total d'exécution de tous les travaux*

I-1.2.1 Cas du même ordre de passage sur les centres de production A et B

- Pb de **flow shop** à 2 centres de production
- **Exemple**

Numéro de la tâche i	1	2	3	4	5	
t_{iA}	50	150	80	200	30	
t_{iB}	60	50	150	70	200	
Rang					1	


- **Algorithme de Johnson**
 - **Étape 1p**: Chercher i dont t_{ij} (avec $j = A$ ou B) est minimum
 - **Étape 2p**:
 - Si $j = A$ placer i à la première place disponible
 - Si $j = B$ placer i à la dernière place disponible
 - **Étape 3p**: Supprimer i des tâches restant à programmer

I-1.2 Ordonnancement de n tâches nécessitant l'intervention de 2 centres de production

- Seul critère repris: *minimisation du temps total d'exécution de tous les travaux*

I-1.2.1 Cas du même ordre de passage sur les centres de production A et B

- Pb de **flow shop** à 2 centres de production
- **Exemple**

Numéro de la tâche i	1	2	3	4	5	
t_{iA}	50	150	80	200	30	
t_{iB}	60	50	150	70	200	
Rang					1	


- **Algorithme de Johnson**
 - **Étape 1p**: Chercher i dont t_{ij} (avec $j = A$ ou B) est minimum
 - **Étape 2p**:
 - Si $j = A$ placer i à la première place disponible
 - Si $j = B$ placer i à la dernière place disponible
 - **Étape 3p**: Supprimer i des tâches restant à programmer

I-1.2 Ordonnancement de n tâches nécessitant l'intervention de 2 centres de production

- Seul critère repris: *minimisation du temps total d'exécution de tous les travaux*

I-1.2.1 Cas du même ordre de passage sur les centres de production A et B

- Pb de **flow shop** à 2 centres de production
- **Exemple**

Numéro de la tâche i	1	2	3	4	5	
t_{iA}	50	150	80	200	30	
t_{iB}	60	50	150	70	200	
Rang					1	


- **Algorithme de Johnson**
 - **Étape 1p:** Chercher i dont t_{ij} (avec $j = A$ ou B) est minimum
 - **Étape 2p:**
 - Si $j = A$ placer i à la première place disponible
 - Si $j = B$ placer i à la dernière place disponible
 - **Étape 3p:** Supprimer i des tâches restant à programmer

I-1.2 Ordonnement de n tâches nécessitant l'intervention de 2 centres de production

- Seul critère repris: *minimisation du temps total d'exécution de tous les travaux*

I-1.2.1 Cas du même ordre de passage sur les centres de production A et B

- Pb de **flow shop** à 2 centres de production
- **Exemple**

Numéro de la tâche i	1	2	3	4	5	
t_{iA}	50	150	80	200	30	
t_{iB}	60	50	150	70	200	
Rang	2				1	


- **Algorithme de Johnson**
 - **Étape 1p**: Chercher i dont t_{ij} (avec $j = A$ ou B) est minimum
 - **Étape 2p**:
 - Si $j = A$ placer i à la première place disponible
 - Si $j = B$ placer i à la dernière place disponible
 - **Étape 3p**: Supprimer i des tâches restant à programmer

I-1.2 Ordonnancement de n tâches nécessitant l'intervention de 2 centres de production

- Seul critère repris: *minimisation du temps total d'exécution de tous les travaux*

I-1.2.1 Cas du même ordre de passage sur les centres de production A et B

- Pb de **flow shop** à 2 centres de production
- **Exemple**

Numéro de la tâche i	1	2	3	4	5	
t_{iA}	50	150	80	200	30	
t_{iB}	60	50	150	70	200	
Rang	2				1	


- **Algorithme de Johnson**
 - **Étape 1p**: Chercher i dont t_{ij} (avec $j = A$ ou B) est minimum
 - **Étape 2p**:
 - Si $j = A$ placer i à la première place disponible
 - Si $j = B$ placer i à la dernière place disponible
 - **Étape 3p**: Supprimer i des tâches restant à programmer

I-1.2 Ordonnement de n tâches nécessitant l'intervention de 2 centres de production

- Seul critère repris: *minimisation du temps total d'exécution de tous les travaux*

I-1.2.1 Cas du même ordre de passage sur les centres de production A et B

- Pb de **flow shop** à 2 centres de production
- **Exemple**

Numéro de la tâche i	1	2	3	4	5	
t_{iA}	50	150	80	200	30	
t_{iB}	60	50	150	70	200	
Rang	2				1	


- **Algorithme de Johnson**
 - **Étape 1p:** Chercher i dont t_{ij} (avec $j = A$ ou B) est minimum
 - **Étape 2p:**
 - Si $j = A$ placer i à la première place disponible
 - Si $j = B$ placer i à la dernière place disponible
 - **Étape 3p:** Supprimer i des tâches restant à programmer

I-1.2 Ordonnancement de n tâches nécessitant l'intervention de 2 centres de production

- Seul critère repris: *minimisation du temps total d'exécution de tous les travaux*

I-1.2.1 Cas du même ordre de passage sur les centres de production A et B

- Pb de **flow shop** à 2 centres de production
- **Exemple**

Numéro de la tâche i	1	2	3	4	5	
t_{iA}	50	150	80	200	30	
t_{iB}	60	50	150	70	200	
Rang	2	5			1	


- **Algorithme de Johnson**
 - **Étape 1p**: Chercher i dont t_{ij} (avec $j = A$ ou B) est minimum
 - **Étape 2p**:
 - Si $j = A$ placer i à la première place disponible
 - Si $j = B$ placer i à la dernière place disponible
 - **Étape 3p**: Supprimer i des tâches restant à programmer

I-1.2 Ordonnancement de n tâches nécessitant l'intervention de 2 centres de production

- Seul critère repris: *minimisation du temps total d'exécution de tous les travaux*

I-1.2.1 Cas du même ordre de passage sur les centres de production A et B

- Pb de **flow shop** à 2 centres de production
- **Exemple**

Numéro de la tâche i	1	2	3	4	5	
t_{iA}	50	150	80	200	30	
t_{iB}	60	50	150	70	200	
Rang	2	5			1	


- **Algorithme de Johnson**
 - **Étape 1p**: Chercher i dont t_{ij} (avec $j = A$ ou B) est minimum
 - **Étape 2p**:
 - Si $j = A$ placer i à la première place disponible
 - Si $j = B$ placer i à la dernière place disponible
 - **Étape 3p**: Supprimer i des tâches restant à programmer

I-1.2 Ordonnement de n tâches nécessitant l'intervention de 2 centres de production

- Seul critère repris: *minimisation du temps total d'exécution de tous les travaux*

I-1.2.1 Cas du même ordre de passage sur les centres de production A et B

- Pb de **flow shop** à 2 centres de production
- **Exemple**

Numéro de la tâche i	1	2	3	4	5	
t_{iA}	50	150	80	200	30	
t_{iB}	60	50	150	70	200	
Rang	2	5			1	


- **Algorithme de Johnson**
 - **Étape 1p:** Chercher i dont t_{ij} (avec $j = A$ ou B) est minimum
 - **Étape 2p:**
 - Si $j = A$ placer i à la première place disponible
 - Si $j = B$ placer i à la dernière place disponible
 - **Étape 3p:** Supprimer i des tâches restant à programmer

I-1.2 Ordonnement de n tâches nécessitant l'intervention de 2 centres de production

- Seul critère repris: *minimisation du temps total d'exécution de tous les travaux*

I-1.2.1 Cas du même ordre de passage sur les centres de production A et B

- Pb de **flow shop** à 2 centres de production
- **Exemple**

Numéro de la tâche i	1	2	3	4	5	
t_{iA}	50	150	80	200	30	
t_{iB}	60	50	150	70	200	
Rang	2	5		4	1	


- **Algorithme de Johnson**
 - **Étape 1p:**
 - **Étape 2p:**
 - Si $j = A$ placer i à la première place disponible
 - Si $j = B$ placer i à la dernière place disponible
 - **Étape 3p:** Supprimer i des tâches restant à programmer

I-1.2 Ordonnancement de n tâches nécessitant l'intervention de 2 centres de production

- Seul critère repris: *minimisation du temps total d'exécution de tous les travaux*

I-1.2.1 Cas du même ordre de passage sur les centres de production A et B

- Pb de **flow shop** à 2 centres de production
- **Exemple**

Numéro de la tâche i	1	2	3	4	5	
t_{iA}	50	150	80	200	30	
t_{iB}	60	50	150	70	200	
Rang	2	5		4	1	


- **Algorithme de Johnson**
 - **Étape 1p:**
 - **Étape 2p:**
 - Si $j = A$ placer i à la première place disponible
 - Si $j = B$ placer i à la dernière place disponible
 - **Étape 3p:** Supprimer i des tâches restant à programmer

I-1.2 tâches nécessitant l'intervention de 2 centres de production

- Seul critère repris: *minimisation du temps total d'exécution de tous les travaux*

I-1.2.1 Cas du même ordre de passage sur les centres de production A et B

- Pb de **flow shop** à 2 centres de production
- **Exemple**

Numéro de la tâche i	1	2	3	4	5	
t_{iA}	50	150	80	200	30	
t_{iB}	60	50	150	70	200	
Rang	2	5	3	4	1	

- **Algorithme de Johnson**

- **Étape 1p:**

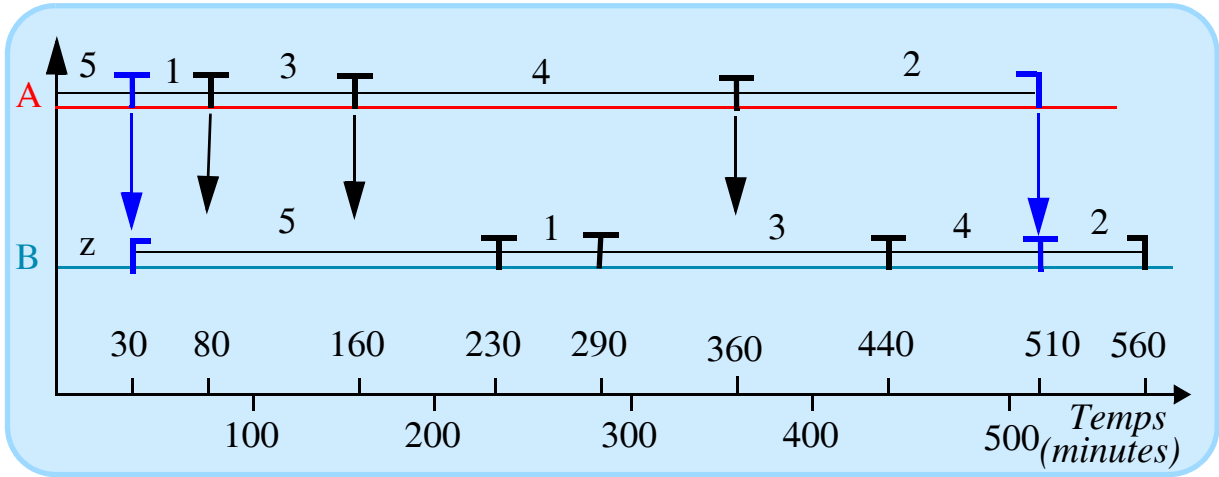
- **Étape 2p:**

- Si $j = A$ placer i à la première place disponible
- Si $j = B$ placer i à la dernière place disponible

- **Étape 3p:** Supprimer i des tâches restant à programmer

Solution optimale 5 – 1 – 3 – 4 – 2

Ordonnement en ateliers spécialisés



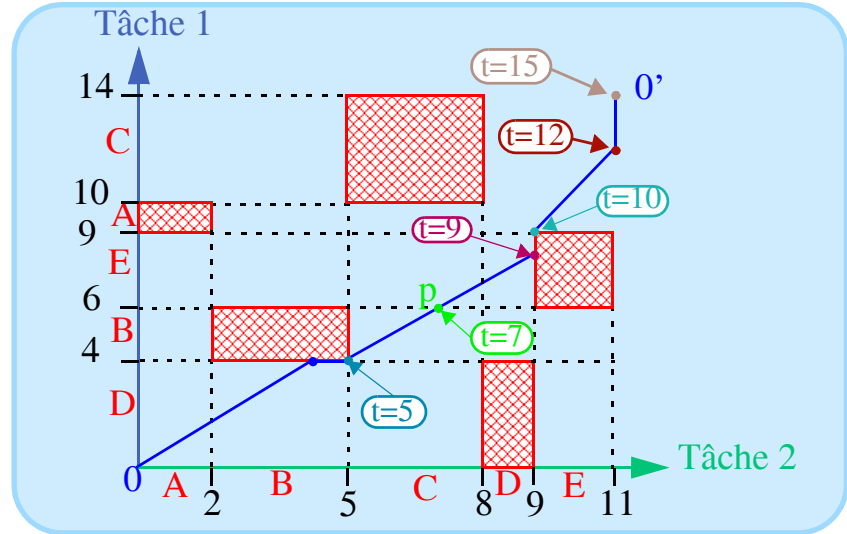
I-1.2.2 Cas de la non-unicité de l'ordre de passage sur les centres de production A et B

• **Algorithme de Jackson**

- **partition** en 4 de l'ensemble initial
 - {A} : toutes les tâches ne nécessitant que l'intervention de A
 - {B} : toutes les tâches ne nécessitant que l'intervention de B
 - {AB} : toutes les tâches passant par A puis B
 - {BA} : toutes les tâches passant par B puis A
- ordonnancement optimal
 - algorithme de Johnson sur {AB} \Rightarrow séquence 1
 - algorithme de Johnson sur {BA} \Rightarrow séquence 2
 - ordre quelconque sur {A} \Rightarrow séquence 3
 - ordre quelconque sur {B} \Rightarrow séquence 4
 - Sur le centre A séquences 1 puis 3 puis 2
 - Sur le centre B séquences 2 puis 4 puis 1

I-1.3 Ordonnement de 2 tâches nécessitant l'intervention de m centres de production

	Tâche			
	1		2	
	TO	rang	TO	rang
A	1	4	2	1
B	2	2	3	2
C	4	5	3	3
D	4	1	1	4
E	3	3	2	5



I-1.4 Ordonnement de n tâches nécessitant l'intervention de m centres de production

I-1.4.1 Ordonnement de n tâches nécessitant l'intervention de 3 centres de production (ordre identique de passage)

- **Application algorithme Johnson** sur A-B-C si $\underset{i}{\text{Max}}(t_{iB}) \leq \underset{i}{\text{Min}}(t_{iA})$ **ou** $\underset{i}{\text{Max}}(t_{iB}) \leq \underset{i}{\text{Min}}(t_{iC})$

sur pb fictif

- machine virtuelle α regroupant A et B $\Rightarrow t_{iAB} = t_{iA} + t_{iB}$
- machine virtuelle γ regroupant A et C $\Rightarrow t_{iAB} = t_{iC} + t_{iB}$

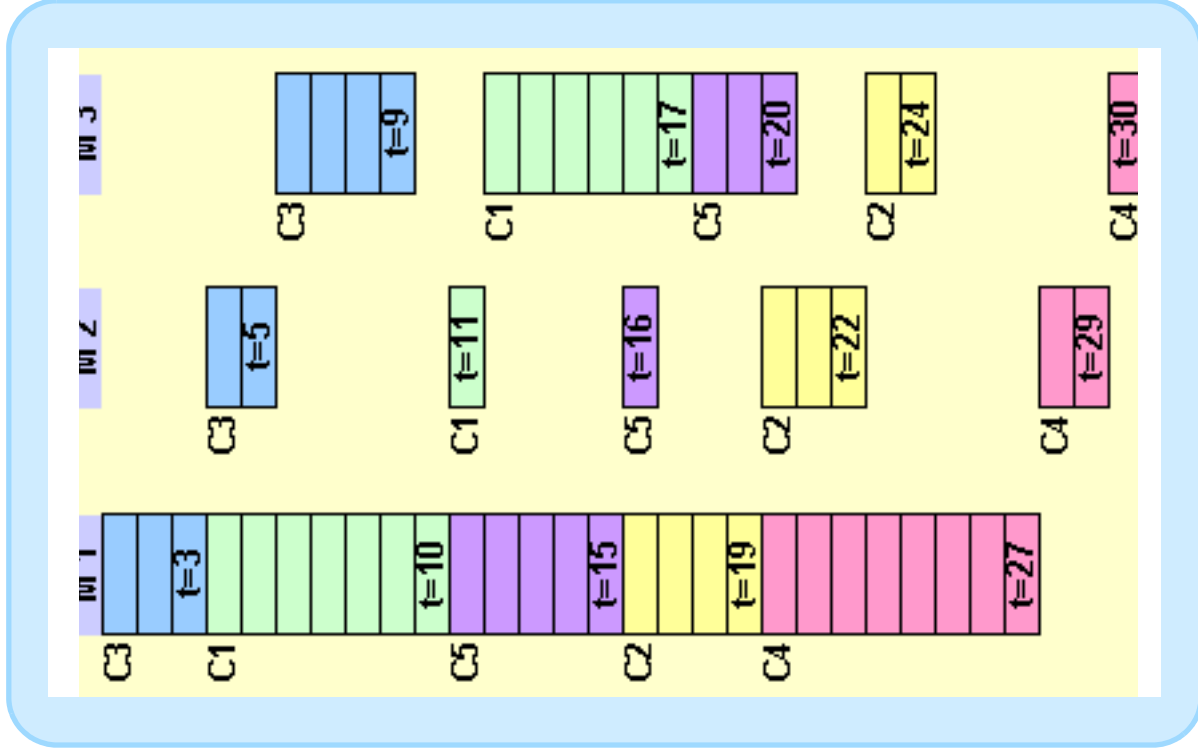
Tâche i	t_{iA}	t_{iB}	t_{iC}
1	7	1	6
2	4	3	2
3	3	2	4
4	8	2	1
5	5	1	3
	$\min t_{iA} = 3$	$\max t_{iB} = 3$	$\min t_{iC} = 1$

Tâche i	t_{iAB}	t_{iBC}
1	8	7
2	7	5
3	5	6
4	10	3
5	6	4



B dominé par A (mais pas par C)

• Solution



I-1.4.2 Ordonnement de n tâches nécessitant l'intervention de m centres de production (ordre identique de passage)

I-1.4.2.1 Le modèle de base

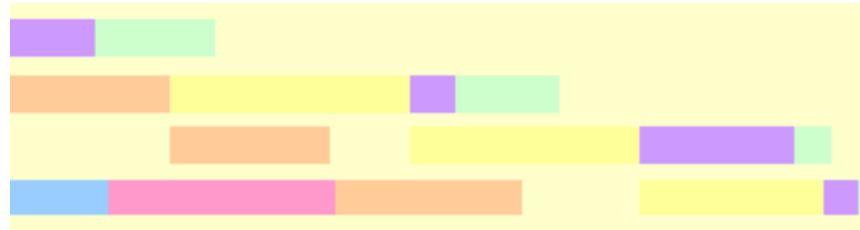
- $(n!)^m$ ordonnements possibles
- Algorithme CDS
 - exemple 5 CP (A à E)

Tâche i	Temps d'exécution en 1/10ème d'heure			
	t_{iA}	t_{iB}	t_{iC}	t_{iD}
1	50	43	15	4
2	89	99	95	77
3	7	47	20	98
4	8	64	12	94
5	61	19	65	14
6	1	80	66	78



- résolution des 4 problèmes suivants:

$$\{A\} - \{E\}; \{AB\} - \{DE\}; \{ABC\} - \{CDE\}; \{ABCD\} - \{BCDE\}$$



I-1.4.2.2 Prise en compte des temps de montage / démontage dépendants de l'ordre de passage des tâches

Pour mémoire

I-1.4.2.3 Ordonnancement de n tâches nécessitant l'intervention de m centres de production (ordre identique de passage – *sans attente*)

Pour mémoire

I-1.4.2.4 *Le flow shop* hybride

Pour mémoire

I-1.5 Ordonnement de n tâches nécessitant l'intervention de m centres de production (cheminement libre – *open shop*)

Pour mémoire

I-1.6 Ordonnement de n tâches nécessitant l'intervention de m centres de production (ordre de passage quelconque)

- Aucun résultat **analytique**
- Démarche **heuristique** si **goulot d'étranglement** : piloter le système en s'appuyant sur un ordonnancement défini pour ce goulot
- **Exemple**

1		2		3		4		5	
Machine	durée	Machine	durée	Machine	durée	Machine	durée	Machine	durée
A	5	A	3	C	8	B	5	D	7
C	7	B	5	A	4	D	4	C	15
D	9	C	10	B	3	C	6	A	4
-	-	D	4	-	-	B	7	-	-

- **Principe**: détermination du goulot / machine fictive avant / machine fictive après ; hyp implicite de capacité infinie avant et après goulot
- **Goulot**: A ($5+3+4+0+4=16$); B ($0+5+3+5+0=13$); **C** ($7+10+8+6+15=46$); D ($9+4+0+4+7=24$)
 - Amont: cumul travail \Rightarrow date d'arrivée (au + tôt) dans goulot
 - Aval: dte de livraison – cumul travail aval = dates de livraison (au + tard) goulot
 - tâches n'utilisant le goulot: fusion avec amont ou traitées à part

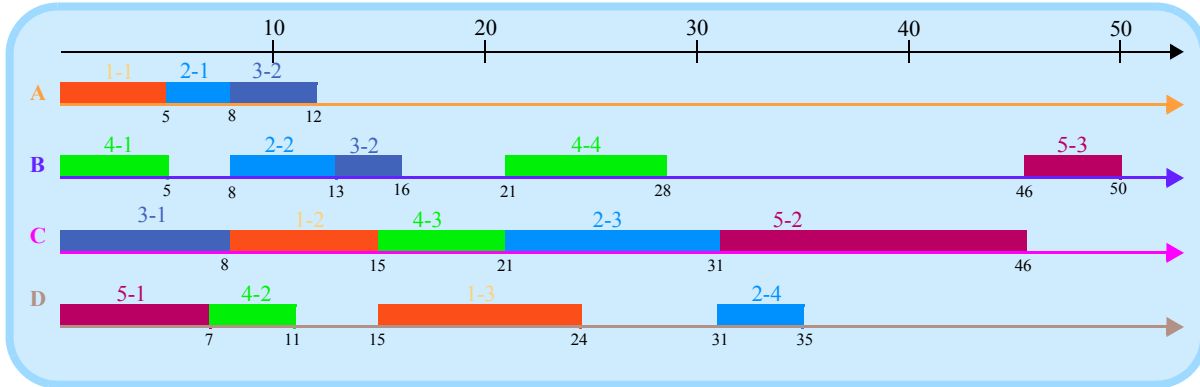
- Application

Tâche 1		Tâche 2		Tâche 3		Tâche 4		Tâche 5	
Machine	durée	Machine	durée	Machine	durée	Machine	durée	Machine	durée
Avant C	5	Avant C	8	Avant C	0	Avant C	9	Avant C	7
C	7	C	10	C	8	C	6	C	15
Après C	9	Après C	4	Après C	7	Après C	7	Après C	4

- Résolution ordo sur goulot (*simple embeded one-resource problem*) ici règle TOM dynamique
 - en T = 0: chargement de la tâche 3 immédiatement disponible (durée 8);
 - en T = 5: arrivée de la tâche 1 (durée 7);
 - en T = 7: arrivée de la tâche 5 (durée 15);
 - en T = 8: fin de la tâche 3, libération de la machine C; arrivée de 2 (durée 10); chargement de 1 (en application de la règle TOM, les tâches 1 et 5 étant candidates);
 - en T = 9: arrivée de 4 (durée 6);
 - en T = 15: fin de la tâche 1, libération de la machine C; chargement de la tâche 4 (en application de la règle TOM, les tâches 4 et 5 étant candidates);
 - en T = 21: fin de 4, libération de la machine C; chargement de la tâche 2 (en application de la règle TOM, les tâches 2 et 5 étant candidates);
 - en T = 31: fin de la tâche 2, libération de la machine C; chargement de la tâche 5 (candidat unique);
 - en T = 46: fin de la tâche 5
- Date de début dans goulot = date de livraison de l'amont et date de sorte du goulot = date d'arrivée de l'aval / règles de priorité locales utilisées en amont et aval (S/OPN...)

Ordonnement en ateliers spécialisés

- Résultat



I-2 Modèles statiques: cas du coût de lancement total variable avec l'ordonnancement retenu

Pour mémoire

I-2.1 Présentation de l'algorithme de Little, Marty, Sweeney & Karel

Pour mémoire

I-2.2 Remarques complémentaires

Pour mémoire

I-2.2.1 Détermination empirique de la tournée

Pour mémoire

I-2.2.2 Détermination optimale de tournées multiples

Pour mémoire

I-2.2.3 Problème stochastique du voyageur de commerce

Pour mémoire

I-2.2.4 Complexité des problèmes concrets

Pour mémoire

I-3 Tentative de caractérisation de l'approche statique

Pour mémoire

I-3.1 Critère d'optimisation

Pour mémoire

I-3.2 Liste des hypothèses décrivant le système productif

Pour mémoire

I-3.3 Méthodes de résolution

Pour mémoire

SECTION II L'APPROCHE ALÉATOIRE DYNAMIQUE

- Variables aléatoires de caractéristiques stables \Rightarrow recherche comportement système (variables d'état caractérisant «régime de croisière») résultant ensemble de règles de décision

II-1 L'approche par la théorie des files d'attente

- **Caractéristiques:**
 - **Arrivées** aléatoires des tâches dans SP
 - **SP** = un ou plusieurs postes de travail, fonctionnant en parallèle ou en série
 - **Loi de service** pour chaque poste de travail
 - **Discipline de file d'attente**
 - Résultats analytiques = $E(\text{variable d'état})$ caractérisant **régime stationnaire**.
- Peu de résultats (configuration très simple)

II-2 L'approche simulateur

- Monte-Carlo pour obtenir info pour systèmes complexes, éventuellement perturbés
- Utilisé à partir des années 60 (coût acceptable) / trentaine de simulateurs disponibles

II-2.1 La simulation de systèmes réels

- Recherche de règles de décision générales (tables de décision) ou contingente (pb spécifique)
- Supériorité de règle: contingente, attention à généralisations abusives

II-2.2 La simulation de systèmes fictifs

- Hypothèses précises: nombre de CP, gammes, durées, arrivées, lotissement, temps de transfert, perturbations...

II-2.2.1 Le cas des ateliers spécialisés indépendants

- Conway, Maxwell et Miller: jeu de 8700 tâches, SP à 9 C, 25 règles de priorité myope), arrivées, gammes
- Temps d'Achèvement Moyen \bar{A} : adapté (E(), arrivées non simultanées)
- Principales règles de priorité testées
 - **RANDOM** (pour étalonnage)
 - **PAPS** Premier-Arrivé, Premier-Servi (*FCFS*); performances moyennes voisines de RANDOM
 - TOM Temps Opérateur Minimum (SPT)
 - **LWKR**, *Least Work Remaining*

Ordonnancement en ateliers spécialisés

- **S/OPN** = quotient de la **marge** (= temps restant avant la livraison, diminué du cumul des temps opératoires restant à réaliser) par le nombre d'opérations restant à exécuter
- **WINQ** (pour *Work in Next Queue*); priorité = S TO tâches en attente + éventuellement TO résiduel de tâche en cours)
- Évaluation dynamique; ne repose pas sur même SI (règles ± myopes)
- **Exemple**: machine A se libérant à l'instant $t = 90'$, charge de travail résiduelle + en attente $F=65, G=100, K=0$

Tâches	Temps opératoire sur la machine A	Opération suivante		Cumul de <i>tous</i> les temps opératoires restant à exécuter à l'instant $t = 90$	Date de livraison demandée	Nombre d'opérations restant à exécuter
		à exécuter sur la machine	temps opératoire			
a	10	K	22	100	$t = 270$	4
b	20	K	10	37	$t = 170$	3
c	17	F	9	41	$t = 270$	2
d	15	G	4	29	$t = 170$	4

- Résultats
 - TOM \Rightarrow a; Winq \Rightarrow a ou b;
 - marge \Rightarrow b ($170 - 37 - 90 = 43$; a=80; c=139; d=51)
 - S/OPN \Rightarrow d

• **Comparaison** des règles.

Règles		RANDOM	FCFS	TOM (SPT)	LWKR	WINQ	SOPN
Nombre moyen instantané de tâches en attente dans le système		59,42	58,87	23,25	47,52	40,43	Données non disponibles
Temps d'achèvement total d'une tâche	x	74,70	74,43	34,02	Données non disponibles		66,10
	σ	Données non disponibles	41,06	53,65			16,31

• **Remarques:**

- **Robustesse** de TOM si erreur $\pm 10\%$
- **Partage** en urgent et non urgent: performance correcte si $< 30\%$ urgent
- TOM retarde **opérations longues** (\Rightarrow bascule périodique sur PEPS)
- Si pas trop engorgé: S/OPT sinon TOM

II-2.2.2 Cas d'une dépendance entre les centres de production

- Performances contingente:
 - indépendance en proba des gammes / pas de structure arborescente des CP
 - polyvalence baisse prédominance de TOM)

SECTION III PERSPECTIVES ACTUELLES DE L'ORDONNANCEMENT EN ATELIERS SPÉCIALISÉS

- Ici SP en ateliers spécialisés, en îlots de fabrication, ou en lignes d'assemblage et/ou de fabrication
- Ordo reste préoccupation (prod à la commande) même si appro/prod synchrone et JAT
- Importance de l'I en procédures (SIAD...) sousestimée par manque de modélisation préalable et qualité SI mais aussi parce que flexibilité physique privilégiée
- SIAD ordo + mobilisation ponctuelle de ressources ? évaluation éco globale des alternatives

III-1 Les approches possibles

III-1.1 Exemple introductif

[Voir Données du problème](#)

III-1.2 Les solutions possibles

III-1.2.1 Placement progressif d'ordres de fabrication

[Voir Placement progressif](#)

III-1.2.2 Placement chronologiquement progressif d'opérations exécutables

[Voir Placement chronologique](#)



III-2 Définition d'un Système Interactif d'Aide à la Décision de Lancement (SIADL)

Pour mémoire