Université Internationale de Casablanca

UNIVERSITÉ RECONNUE PAR L'ÉTAT

Université Internationale de Casablanca

Ecole d'Ingénierie

Filière : Génie Industriel

Ordonnancement de la production

Année Universitaire : 2019-2020

Plan du cours

Introduction

I. Description des problèmes d'ordonnancement

1. Objectifs de l’ordonnancement
2. Données d’un problème d’ordonnancement

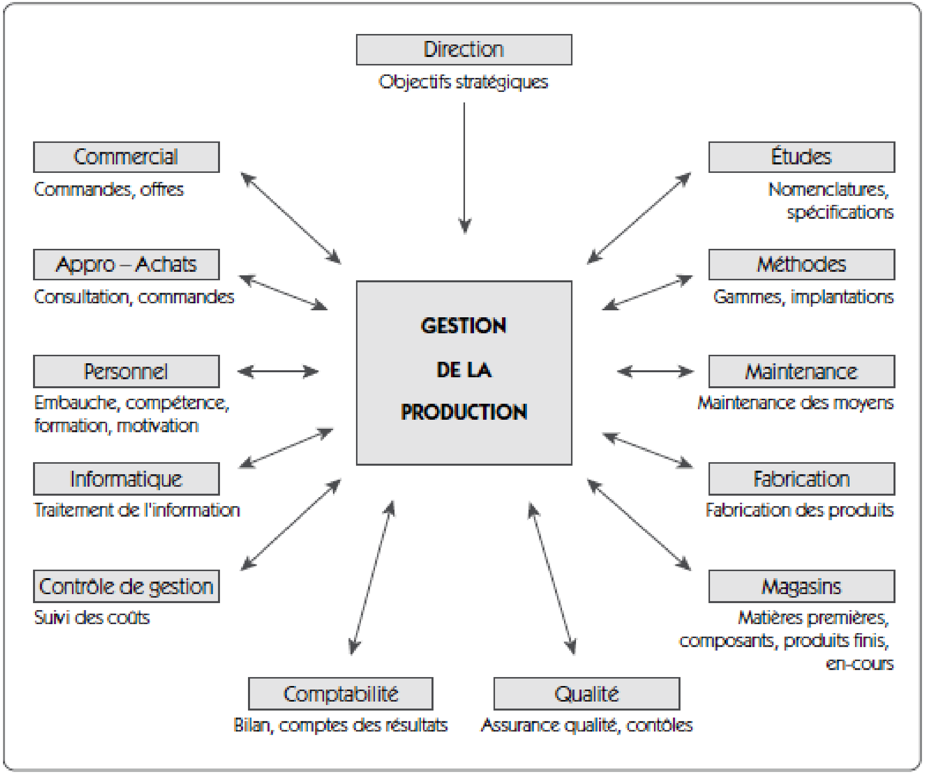
II. Classification des problèmes d'ordonnancement

1. Types de problèmes
2. Schémas de classification

III. Méthodes de résolution des problèmes d’ordonnancement

1. Méthodes de résolution approchées
2. Méthodes de résolution exactes

Introduction



Le processus de planification : la planification hiérarchisée

Niveau 0 : Plan stratégique à très long terme

Nécessité de prévisions pour disposer des ressources nécessaires à la fabrication

Niveau 1 : Plan directeur
Planification de la capacité

Prévisions à long terme pour ajustements majeurs des capacités de production

Niveau 2 : Gestion des flux
Calcul des besoins et gestion des stocks

Prévisions à moyen terme pour ajustements mineurs des capacités et passation de commandes aux fournisseurs

Niveau 3 : Ordonnancement
Gestion des priorités

Prévisions à court terme pour réaffectation des ressources et gestion des priorités

Le processus de planification : la planification hiérarchisée

- Recouvre l'ensemble des décisions prises dans les domaines :
  - Commercial, financier, production.

- **Moyen terme (une année) - Plan de production**

  - Plan Industriel et Commercial -

  - Ajustements de la capacités par rapport à la demande

- **Moyen terme (1 à 3 mois) - Programme Directeur de Production (PDP)**
  - Détermine pour chaque article les quantités à produire.

- **Plan de besoin en matières (Calcul des besoins nets)**
  - Détermine les matières premières à commander et les composants à produire.

- **Court terme - Ordonnancement de la production**
  - Affecter les ressources aux différentes opérations.

Qu'est ce que l'ordonnancement ?

Ordonnancement : la détermination de l'ordre de traitement des commandes en indiquant pour chaque tâche à exécuter où et à quel moment elle sera effectuée.

Jalonnement : détermination de la séquence selon laquelle les tâches seront effectuées par un poste de travail.

Etapes :

1. L'affectation : distribution des tâches aux postes de travail
2. Détermination d'un ordre de passage : détermination de la séquence de traitement des commandes à chaque poste de travail: jalonnement.
3. Calendrier de fabrication : date et heure de lancement des opérations à chaque poste de travail.
4. Lancement : démarrage des opérations selon le calendrier.
5. Suivi : supervision de l'exécution et vérification de l'adéquation avec la planification.
6. Relance : ajustements en fonction des imprévus.

Objectifs d'ordonnancement

1. Rencontrer les dates promises ;
2. Minimiser les en-cours ;
3. Minimiser le temps moyen de passage à travers le système (atelier) ;
4. Minimiser les temps d'arrêts ;
5. Réduire les temps de mise en place ;
6. Minimiser les coûts.

1. Objectifs de l'ordonnancement

- **Objectifs liés au temps**

Minimisation du temps total d'exécution, du temps moyen d'achèvement, des durées totales de réglage ou des retards par rapport à la date de livraison, etc.

- **Objectifs liés aux ressources**

Maximiser la charge d'une ressource, minimiser le nombre de ressources nécessaires pour réaliser un ensemble de tâches, etc.

- **Objectifs liés aux coûts**

Minimiser les coûts de lancement, de production, de stockage, de transport, etc.

2. Données d’un problème d’ordonnancement

Les Tâches

•Une tâche est définie par un ensemble de variables où chacune est une information particulière se rapportant à la réalisation d’une opération.

•Un ensemble de tâches dans un ordre précis forme un job pour lequel on donne la gamme de production.

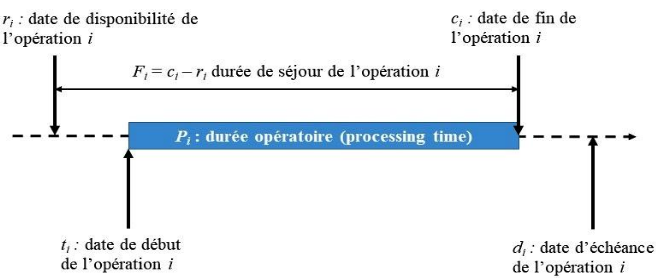
•**Les tâches préemptibles :** le traitement de ces dernières peut être interrompu. Par conséquent, ce type de tâche peut être exécutée en plusieurs fois, facilitant ainsi la résolution de certains problèmes.

•**Les tâches non préemptibles :** elles sont exécutées en une seule fois et leur traitement ne peut être interrompu qu’une fois la tâche terminée.

2. Données d'un problème d'ordonnancement

Les Tâches

- r_i pour représenter la date de disponibilité de la tâche i (release date) ;
- t_i pour représenter la date de début de la tâche i (start date) ;
- c_i pour représenter la date de fin d'exécution de la tâche i (completion time) ;
- d_i pour représenter la date d'échéance de la tâche i (due date) ;
- p_i pour représenter la durée opératoire de la tâche i (processing time date) ;
- $F_i = c_i - r_i$ pour représenter la durée de séjour de l'opération i sur la machine avant qu'elle redevienne disponible (flow time) ;
- $L_i = c_i - d_i$ exprime le retard algébrique (lateness) entre la fin d'exécution de la tâche i par rapport à sa date d'échéance ;
- $T_i = \max(L_i, 0)$ qui exprime le retard absolu de la tâche i (tardiness) ;
- $E_i = \max(-L_i, 0)$ qui exprime l'avancement (earliness) de la tâche i ;



Les Contraintes

Contraintes temporelles :

•**Les contraintes temporelles** intègrent en général les contraintes de temps alloué, issues généralement d’impératifs de gestion et relatives aux dates limites des tâches (tels que les délais de livraison) où a la durée totale d’un projet.

•**Les contraintes d’antériorité** et plus particulièrement les contraintes de cohérence technologique décrivant le positionnement relatif de certaines tâches par rapport à d’autres (e.g contraintes de gammes dans le cas des problèmes d’ateliers).

•**Les contraintes de calendrier** liées au respect d’horaires de travail, etc.

2. Données d'un problème d'ordonnancement

Les Contraintes

Contraintes de ressources :
Ces contraintes sont liées directement à la ressource. Elles spécifient la capacité et la disponibilité des ressources.
Les ressources ne disposent ni de la même disponibilité ni de la même capacité, celles-ci pouvant être modulée par la modification des calendriers d'utilisation ou l'emploi de ressources externes. Une ressource peut aussi être consommable, si à sa libération, elle n'est pas disponible en même quantité.

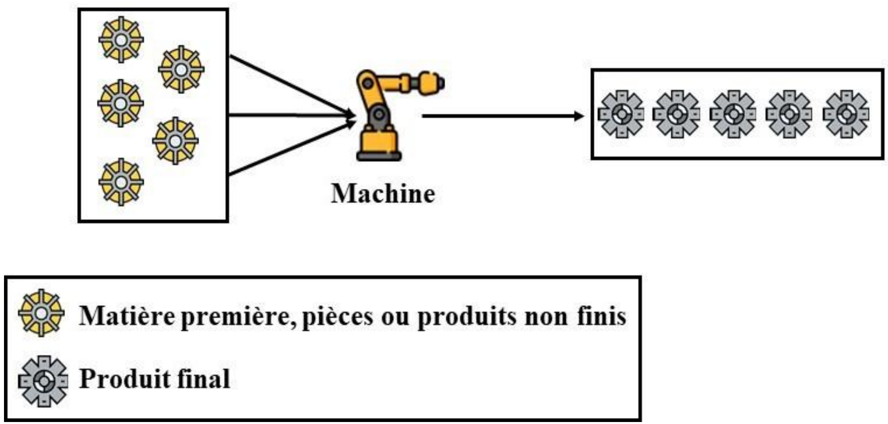
•**Les contraintes endogènes :** elles représentent les contraintes en relation directe avec le système de production, à savoir : la capacité des machines et des moyens de transports, les dates de disponibilités des machines et des moyens de transport, les séquences des opérations (gammes des produits).

•**Les contraintes exogènes :** elles regroupent les contraintes imposées par l'environnement extérieur du système de production : Les dates d'échéance des produits imposées par le client ou par la nature du produit, les priorités des produits/commandes, les retards possibles permis pour certains produits/commandes.

1. Types de problèmes

Atelier à machine unique

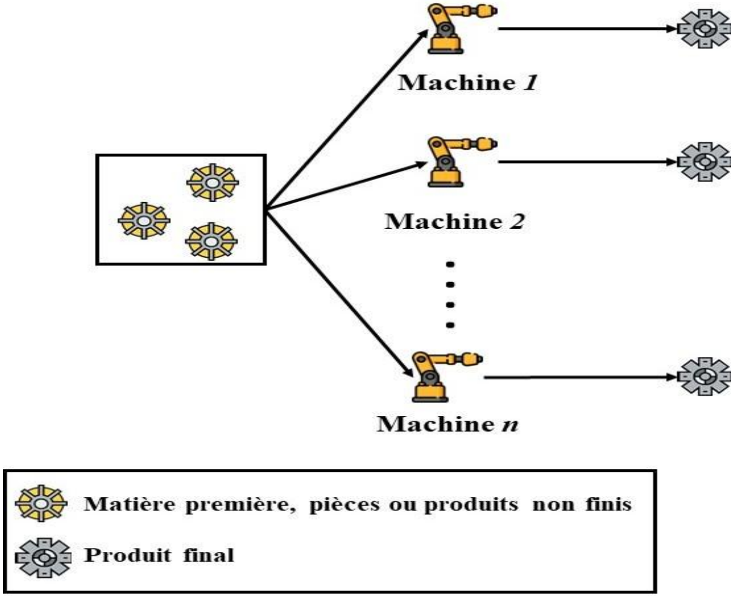
Ce type d'atelier est composé d'une seule machine qui traite toutes les gammes de production. Ce type d'atelier est le plus problématique et est de moins en moins répandu dans l'industrie en raison des différents handicaps rencontrés (goulot d'´etranglement, surexploitation de la ressource, file d'attente interminable, temps de production exponentiel, etc).



1. Types de problèmes

Atelier à machines parallèles

Dans ce type d'atelier, chaque job a la même séquence d'opération. L'atelier à machines parallèles peut être considéré comme étant un atelier à machine unique avec redondance de machines étant donné que toutes les machines sont identiques. Cependant, contrairement au type précédent, la redondance de machine permet d'accélérer la production et d'´eviter l'arrêt du système dans le cas d'une panne de machine.

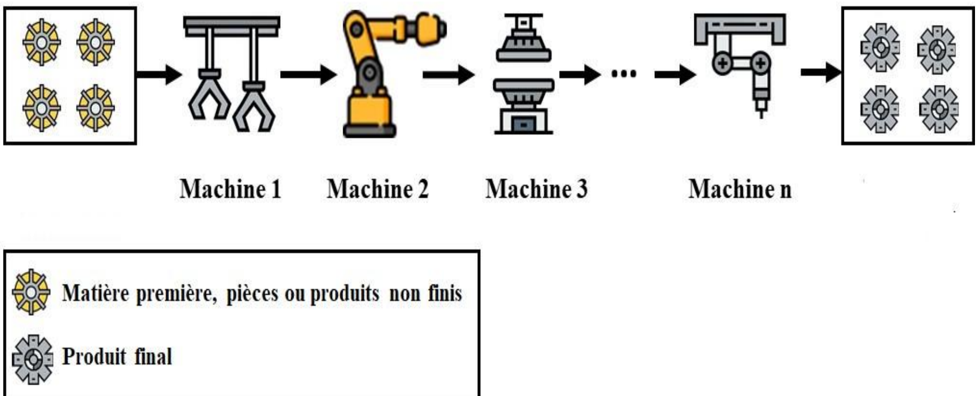


II. Classification des problèmes d'ordonnancement

1. Types de problèmes

Atelier à cheminement unique : Flow shop

Les ateliers de type Flow shop ont pour particularité d'avoir un processus de fabrication linéaire. Ce type d'atelier est constitué d'un ensemble de ressources où le traitement de chaque produit se fait de manière chainée. Le flux de chaque produit est unidirectionnel et chaque opération de chaque job est exécutée dans le même ordre. Ce type d'atelier est rencontré généralement dans les productions en série, où les gammes opératoires sont identiques.

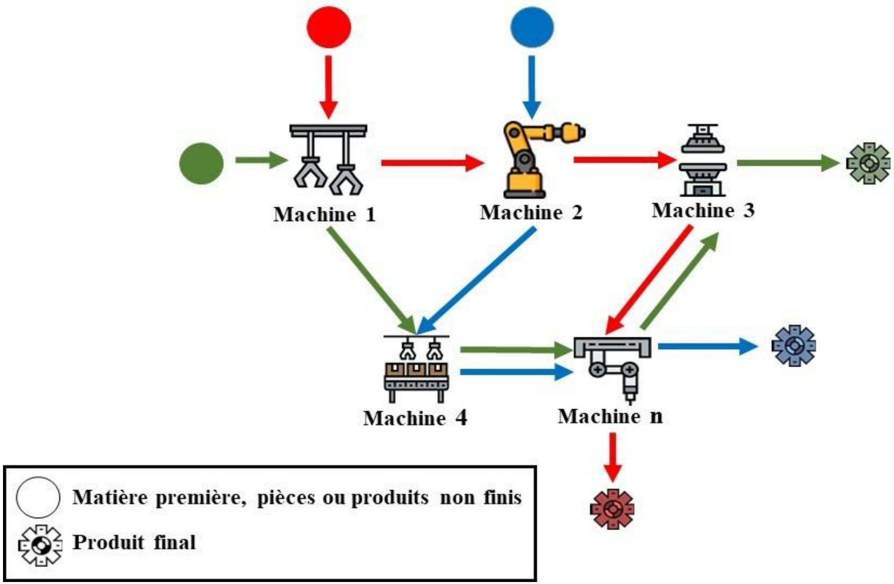


1. Types de problèmes

Atelier à cheminements multiples : Job shop

Dans ce type, les tâches ne s'exécutent pas sur toutes les machines dans le même ordre. En effet, chaque tâche emprunte un chemin qui lui est propre.

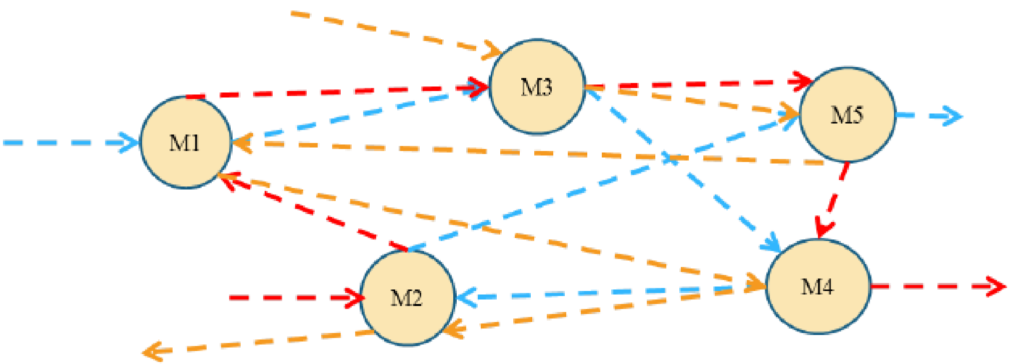
Ce type correspond généralement à une production par lot, notamment dans une unité de production disposant de moyens polyvalents utilisés suivant des séquences différentes afin de réaliser des produits divers.



1. Types de problèmes

Atelier à cheminements libres : Open shop

Dans ce type, les contraintes de précédence sont relâchées. En d’autres termes, les opérations nécessaires à la réalisation de chaque tâche peuvent être effectuées dans n’importe quel ordre (les gammes sont libres).



2. Schémas de classification

Système de notation *(α | β | γ)*

(Classification de Graham)

Champ		Sous champs	Notation
α	(α_1)	Type de machine	$\{\emptyset, 1, P, Q, R, F, J, O, FH, JF, OG\}$
	(α_2)	Nombre de machines	$\{\emptyset, m\}$
β	(β_1)	Mode d'exécution des jobs	$\{\emptyset, pmtn\}$
	(β_2)	Ressources supplémentaire	$\{\emptyset, res\}$
	(β_3)	Relation de précedence	$\{\emptyset, prec, tree, cahins\}$
	(β_4)	Dates de disponibilité	$\{\emptyset, r_i\}$
	(β_5)	Durées opératoires	$\{\emptyset, p_i = p\}$
	(β_6)	Dates d'échéance	$\{\emptyset, d_i\}$
	(β_7)	Propriété d'attente	$\{\emptyset, nwt\}$
γ		/	$\{C_{max}, \sum w_i C_i, L_{max}, T_{max}, \sum w_i T_i, \sum U_i, \sum w_i U_i\}$

2. Schémas de classification

Système de notation $(\alpha \mid \beta \mid \gamma)$

Description du champ α_1

Notation	Description
1	Problème à une seule machine
P	Problème à machines parallèles identiques
Q	Problème à machines parallèles uniformes
R	Problème à machines parallèles indépendantes
F	Flow-Shop
J	Job-Shop
O	Open-Shop
FH	Flow-Shop hybride
JF	Job-Shop flexible
OG	Open-Shop généralisé

2. Schémas de classification

Système de notation (α | β | γ)

Description du champ β

Notation	Description
$pmtn$	La préemption des opérations est autorisée
$prec$	Existence des contraintes de précedence entre les opérations
res	L'opération nécessite l'emploi d'une ou plusieurs ressources supplémentaires
nwt	Les opérations de chaque job doivent se succéder sans attente
$p_i = p$	Les temps d'exécution des tâches sont identiques et égaux à p
r_i	Une date de début au plus tôt est associée à chaque job i
d_i	Une date d'échéance est associée à chaque job i

2. Schémas de classification

Système de notation $(\alpha \mid \beta \mid \gamma)$

Description du champ γ

Notation	Expression	Description
C_{max}	$\max_{i\in\{1,\dots,n\}} C_i$	La durée totale de l'ordonnancement
L_{max}	$\max_{i\in\{1,\dots,n\}} C_i - d_i$	Le plus grand retard algébrique
T_{max}	$\max_{i\in\{1,\dots,n\}}\{\max(C_i - d_i, 0)\}$	Le plus grand retard vrai
$\sum[w_i]C_i$	$\sum_{i\in\{1,\dots,n\}}[w_i]C_i$	La somme [pondéré] des dates de fin des tâches
$\sum[w_i]T_i$	$\sum_{i\in\{1,\dots,n\}}[w_i]\{\max(C_i - d_i, 0)\}$	La somme [pondéré] des retards
$\sum[w_i]U_i$	$\sum_{i\in\{1,\dots,n\}}[w_i]\lvert\{J_i / C_i > d_i\}\rvert$	Le nombre [pondéré] des tâches en retard

2. Schémas de classification

Système de notation $(\alpha \mid \beta \mid \gamma)$

Exemples :

$\mathbf{J_2 \| \| C_{max}}$	dénote un problème d'ordonnancement d'un atelier de type Job shop à 2 machines avec minimisation de makespan C_{max}. L'absence de valeurs dans le champ β.
$\mathbf{F_4 \| \| C_{max}}$	dénote un problème d'ordonnancement d'un atelier de type Flow shop à 4 machines avec minimisation de makespan C_{max}. L'absence de valeurs dans le champ β.
$\mathbf{P_m \| pmtn \| L_{max}}$	désigne le problème de la minimisation du retard maximum L_{max} dans un environnement à m machines parallèles identiques où la préemption est autorisée.
$\mathbf{1 \| prec,r_i \| \Sigma w_i C_i}$	il s'agit d'un problème à une machine dont les tâches présentent une contrainte de précédence et elles ne sont disponibles qu'à la date r_i. Le but est de minimiser la somme pondérée des dates de fin des tâches.

1. Méthodes de résolution approchées

Les Heuristiques

Les heuristiques sont des méthodes empiriques qui donnent généralement de bons résultats sans pour autant être démontrables. Elles se basent sur des règles simplifiées pour optimiser un ou plusieurs critères. Le principe général de cette catégorie de méthodes est d'intégrer des stratégies de décision pour construire une solution proche de celle optimale tout en cherchant à avoir un temps de calcul raisonnable.

- Ordonnancement d'atelier à machine unique
- Ordonnancement d'atelier Flow shop
- Ordonnancement d'atelier Job shop
- Ordonnancement d'atelier Open shop

1. Méthodes de résolution approchées

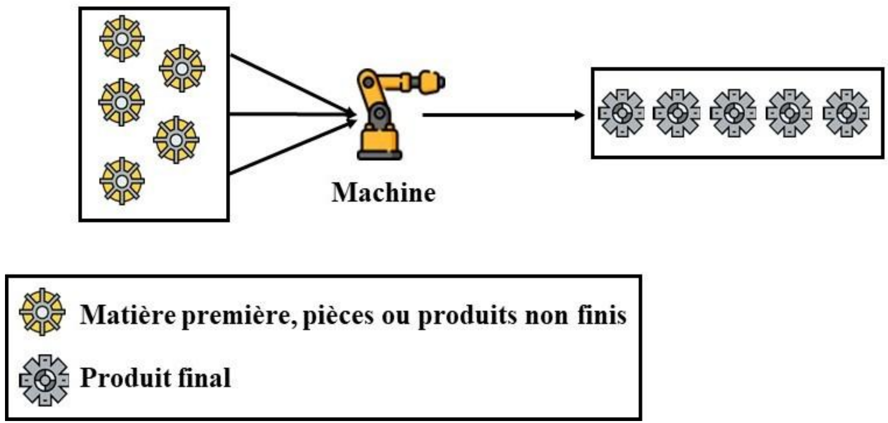
Les Heuristiques

Les heuristiques sont des méthodes empiriques qui donnent généralement de bons résultats sans pour autant être démontrables. Elles se basent sur des règles simplifiées pour optimiser un ou plusieurs critères. Le principe général de cette catégorie de méthodes est d'intégrer des stratégies de décision pour construire une solution proche de celle optimale tout en cherchant à avoir un temps de calcul raisonnable.

- **Ordonnancement d'atelier à machine unique**
- Ordonnancement d'atelier Flow shop
- Ordonnancement d'atelier Job shop
- Ordonnancement d'atelier Open shop

Les Heuristiques

1. Ordonnancement d'atelier à machine unique



Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Règles d'ordonnancement

1. Premier arrivé premier servi (PAPS = FIFO) : les tâches sont traitées dans l'ordre d'arrivée sans aucune exception.

2. Temps de traitement le plus court (shortest processing time) : les tâches sont effectuée selon leurs durées en débutant par la plus courte.

3. La date promise la plus tôt (earliest due date) : les tâches sont effectuées dans l'ordre promis de livraison en débutant par la plus tôt à livrer.

4. Ratio critique (critical ratio) : on calcule le ratio du temps de traitement d'une tâche sur le temps restant avant la date promise. Les tâches seront effectuées dans l'ordre décroissant du ratio.

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Application :

Un centre d'usinage a 5 tâches à réaliser, décider l'heuristique optimal en calculant le retard moyen, le nombre de tâches en retard et le temps moyen dans le système.

Tâche	Temps de traitement	Date promise
1	11	61
2	29	45
3	31	31
4	1	33
5	2	32

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Application :

Règle 1 : FIFO

Tâche	Temps de traitement	Date promise	Date de fin	Df-Dp
1	11	61	11	-50
2	29	45	40	-5
3	31	31	71	40
4	1	33	72	39
5	2	32	74	42

RM : Retard moyen : (40+39+42)/5 = 24.2

TMS : Temps moyen dans le système : (11+40+71+72+74)/5 = 53.6

NbR : Nombre de tache en retard : 3

Application :

Règle 2 : Temps de traitement le plus court (shortest processing time)

Tâche	Temps de traitement	Date promise	Date de fin	Df-Dp
4	1	33	1	-32
5	2	32	3	-29
1	11	61	14	-47
2	29	45	43	-2
3	31	31	74	43

RM : Retard moyen : 43/5= 8.6

TMS : Temps moyen dans le système : (1+3+14+43+74)/5 = 27

NbR : Nombre de tache en retard : 1

Application :

Règle 3 : La date promise la plus tôt (earliest due date)

Tâche	Temps de traitement	Date promise	Date de fin	Df-Dp
3	31	31	31	0
5	2	32	33	1
4	1	33	34	1
2	29	45	63	18
1	11	61	74	13

RM : Retard moyen (1+1+18+13)/5= 6.6

TMS : Temps moyen dans le système (31+33+34+63+74)/5= 47

NbR : Nombre de tache en retard :4

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Application :

Règle 4 : Ratio critique

Nous sommes à t= 0 :

Tâche	Temps de traitement	Date promise	Tps restant	Ratio critique =
1	11	61	61 – 0 = 61	11/61=0.18
2	29	45	45	29/45 =0.64
3	31	31	31	31/31 =1
4	1	33	33	1/33 =0.03
5	2	32	32	2/32 =0.0625

Itération t = t+ T_4

Tâche	Temps de traitement	Date promise	Tps restant	Ratio critique =
1	11	61	61 – 1 = 60	11/60=0.183
2	29	45	44	29/44 =0.659
3	31	31	30	31/30 =1.03
5	2	32	31	2/31 = 0.0645

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Application :

Règle 4 : Ratio critique

Tâche	Temps de traitement	Date promise	Date de fin	Df-Dp
4	1	33	1	-32
5	2	32	3	-29
1	11	61	14	-47
2	29	45	43	-2
3	31	31	74	43

RM : Retard moyen : 43/5= 8.6
TMS : Temps moyen dans le système : (1+3+14+43+74)/5 = 27
NbR : Nombre de tache en retard : 1

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Application :

Méthode	Critère 1 (RM)	Critère 2 (TMS)	Critère 3 (Nb R)
A. Premier arrivé premier servi	24.2	53.6	3
B. Temps de traitement le plus court	27	8.6	1
C. La date promise la plus tôt	47	6.6	4
D. Ratio critique	27	8.6	1
	A - (B,D) - C	C - (B,D) - A	(B,D) - A - C

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

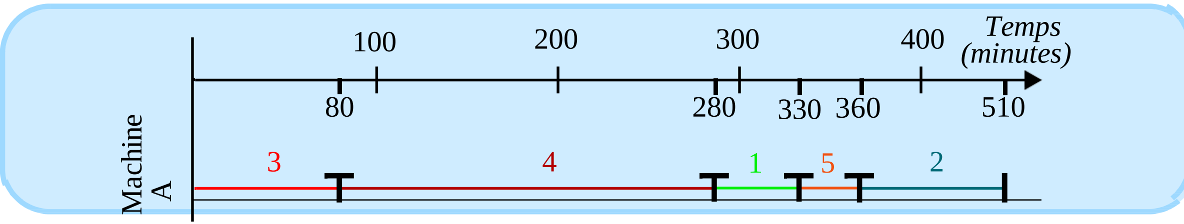
Temps Opératoire Minimum (TOM)

Exemple :

Tâche *i*	1	2	3	4	5
Temps opératoire t_i (en minutes)	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 $\mathbf{T}_j$	80	200	50	30	150
Date $\mathbf{A}_j$ de fin de la tâche *j*	80	280	330	360	510



Les Heuristiques

1. Ordonnancement d’atelier à machine unique

Temps Opératoire Minimum (TOM)

- Dans cet exemple : 5! ordonnancements possibles
- Date d’achèvement : $A_j = \sum_{h=1}^{j} T_h$
- Date moyenne d’achèvement $\bar{A}$

<table>
<tr><td colspan="4">Tâche i</td><td>1</td><td colspan="2">2</td><td colspan="2">3</td><td>4</td><td>5</td></tr>
<tr><td colspan="4">Temps opératoire t_i (en minutes)</td><td>50</td><td colspan="2">150</td><td colspan="2">80</td><td>200</td><td>30</td></tr>
<tr><td rowspan="4"></td><td>Ordre de passage j</td><td>1</td><td>2</td><td>3</td><td>4</td><td colspan="2">5</td><td colspan="3" rowspan="4">$\bar{A} = \frac{1}{5}\sum_{j=1}^{5} A_j = 312$</td></tr>
<tr><td>Tâche programmée j</td><td>3</td><td>4</td><td>1</td><td>5</td><td colspan="2">2</td></tr>
<tr><td>Temps d’exécution $\mathbf{T_j}$</td><td>80</td><td>200</td><td>50</td><td>30</td><td colspan="2">150</td></tr>
<tr><td>Date $\mathbf{A_j}$ de fin de la tâche j</td><td>80</td><td>280</td><td>330</td><td>360</td><td colspan="2">510</td></tr>
</table>

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Temps Opératoire Minimum (TOM)

- **Règle** d'ordonnancement **TOM minimise** $\overline{\mathbf{A}}$

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

- **Application**

Tâche *i*	1	2	3	4	**5**
Temps opératoire t_i (en minutes)	50	150	80	200	**30**

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

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Temps Opératoire Minimum (TOM)

- **Règle** d'ordonnancement **TOM minimise $\overline{A}$**

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

- **Application**

Tâche *i*	**1**	2	3	4	5
Temps opératoire t_i (en minutes)	**50**	150	80	200	30

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

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Temps Opératoire Minimum (TOM)

- **Règle** d'ordonnancement **TOM minimise $\overline{A}$**

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

- **Application**

Tâche i	1	2	3	4	5
Temps opératoire t_i (en minutes)	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**		

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Temps Opératoire Minimum (TOM)

- Règle d'ordonnancement **TOM minimise $\overline{A}$**

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

- **Application**

Tâche i	1	2	3	4	5
Temps opératoire t_i (en minutes)	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	

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Temps Opératoire Minimum (TOM)

- **Règle** d'ordonnancement **TOM minimise** $\bar{A}$

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

- **Application**

Tâche i	1	2	3	**4**	5
Temps opératoire t_i (en minutes)	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**

$\bar{A} = 218$

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Temps Opératoire Minimum (TOM)

- **Règle** d'ordonnancement **TOM 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 minutes)	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**:
  - TOM minimise **retard algébrique moyen: retard algébrique** $(T_j - d_j) \neq$ **retard vrai** $\mathrm{Max}(0, T_j - d_j)$

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Règle TOM pondéré

- Importance : article en rupture de stock, commandes urgentes, etc.
- Pondération u_i ($u_i \geq 1$) traduisant priorité accordée à *i*

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

Règle de Smith : $\frac{T_1}{u_1} \leq \frac{T_2}{u_2} \leq \cdots \leq \frac{T_j}{u_j} \leq \frac{T_{j+1}}{u_{j+1}} \leq \cdots \leq \frac{T_n}{u_n}$

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Exemple

Règle TOM pondéré

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					
$T_h \cdot u_h$					
$\sum_{h=1}^{j} T_h \cdot u_h$					

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Exemple

Règle TOM pondéré

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**				

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Exemple

Règle TOM pondéré

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**			

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Exemple

Règle TOM pondéré

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**		

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Exemple

Règle TOM pondéré

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**	

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Exemple

Règle TOM pondéré

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**

$\bar{A} = 422$

Règle de la date de livraison minimale

Tâche *i*	1	2	3	4	5
Date de livraison d_i souhaitée (en minutes)	100	300	410	400	200
Temps opératoire t_i (en minutes)	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** ordonnançant par dates ↑ de livraison $d_1 \leq d_2 \leq \ldots \leq d_j \leq d_{j+1} \leq \ldots \leq d_n$

Règle de la date de livraison minimale

- **Application.**

Tâche i	1	2	3	4	5
Date de livraison d_i souhaitée (en minutes)	**100**	300	410	400	200
Temps opératoire t_i (en minutes)	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**				

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Règle de la date de livraison minimale

- **Application.**

Tâche *i*	1	2	3	4	5
Date de livraison d_i souhaitée (en minutes)	100	300	410	400	**200**
Temps opératoire t_i (en minutes)	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**			

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Règle de la date de livraison minimale

Application.

Tâche *i*	1	2	3	4	5
Date de livraison d_i souhaitée (en minutes)	100	**300**	410	400	200
Temps opératoire t_i (en minutes)	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**		

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Règle de la date de livraison minimale

- **Application**.

Tâche *i*	1	2	3	4	5
Date de livraison d_i souhaitée (en minutes)	100	300	410	**400**	200
Temps opératoire t_i (en minutes)	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**	

Les Heuristiques

1. Ordonnancement d'atelier à machine unique

Règle de la date de livraison minimale

- **Application.**

Tâche *i*	1	2	3	4	5
Date de livraison d_i souhaitée (en minutes)	100	300	**410**	400	200
Temps opératoire t_i (en minutes)	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
$\bar{A} = 260$

- Remarque: règle de Jackson **minimise** retard max

Règle de la date de la marge minimale

- Ordonnancement par valeurs croissantes de marges ($d_i - t_i$)

$\Rightarrow d_1 - T_1 \leq d_2 - T_2 \leq \ldots \leq d_j - T_j \leq d_{j+1} - T_{j+1} \leq \ldots \leq d_n - T_n$

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

Ordre de passage *j*	1	2	3	4	5
$d_j - T_j$	50	150	330	200	170
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

$\bar{A} = 284$

1. Méthodes de résolution approchées

Les Heuristiques

Les heuristiques sont des méthodes empiriques qui donnent généralement de bons résultats sans pour autant être démontrables. Elles se basent sur des règles simplifiées pour optimiser un ou plusieurs critères. Le principe général de cette catégorie de méthodes est d’intégrer des stratégies de décision pour construire une solution proche de celle optimale tout en cherchant à avoir un temps de calcul raisonnable.

- Ordonnancement d’atelier à machine unique
- **Ordonnancement d’atelier Flow shop**
- Ordonnancement d’atelier Job shop
- Ordonnancement d’atelier Open shop

Algorithme de Johnson

- Cas de 2 machines A et B :
- Ordre identique de passage des tâches sur les machines A et B
- Objectif : minimiser C_{max}

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

Algorithme de Johnson

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

Algorithme de Johnson

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

Algorithme de Johnson

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

Algorithme de Johnson

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

Algorithme de Johnson

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

Algorithme de Johnson

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

Algorithme de Johnson

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

Algorithme de Johnson

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 1**þ: Chercher *i* dont t_{ij} (avec j = A ou B) est minimum
- **Étape 2**þ:
  - Si j = A placer *i* à la première place disponible
  - <u>Si j = B placer *i* à la dernière place disponible</u>
- **Étape 3**þ: Supprimer *i* des tâches restant à programmer

Algorithme de Johnson

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

Algorithme de Johnson

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

Algorithme de Johnson

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

Algorithme de Johnson

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

Algorithme de Johnson

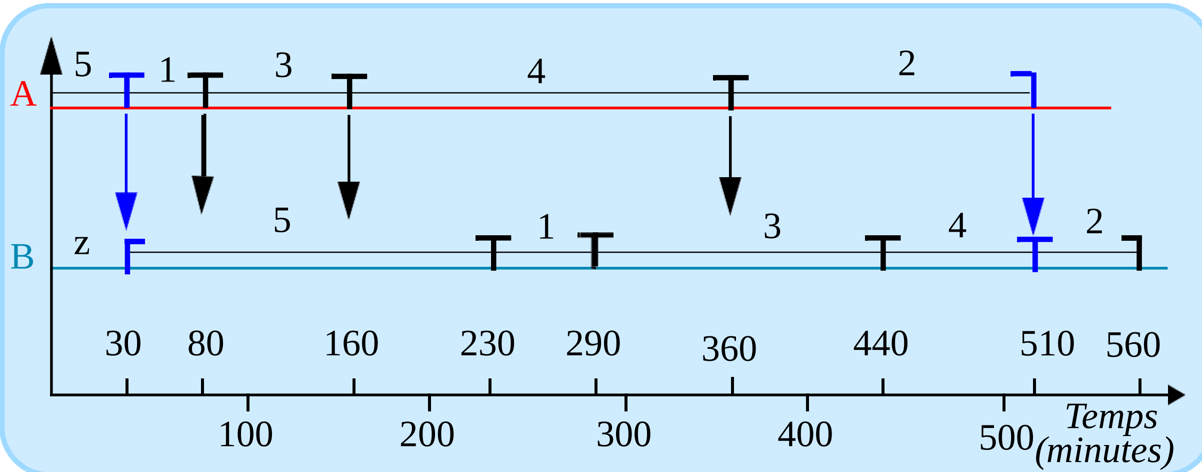
Numéro de la tâche *i*	1	2	3	4	5
t_{iA}	**50**	150	80	200	**30**
t_{iB}	60	**50**	150	**70**	200
Rang	**2**	**5**	**3**	**4**	**1**

Algorithme de Johnson

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

Solution optimale : 5 – 1 – 3 – 4 – 2

Algorithme de Johnson



Algorithme de Johnson

- Cas de 3 machines

- Ordre identique de passage

Application de l'algorithme Johnson sur A-B-C si : $Max(t_{iB}) \le Min(t_{iA})$ **ou** $Max(t_{iB}) \le Min(t_{iC})$

→ Création de 2 machines virtuelles :

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

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

Algorithme de Johnson

- Cas de 3 machines
- Ordre identique de passage

Application :

Tâches	1	2	3	4	5	6	7
Assemblage	20	12	19	16	14	12	17
Inspection	4	1	9	12	5	7	8
Expédition	7	11	4	18	18	3	6

→ On reformule le problème en un problème à deux machines.

Algorithme de Johnson

Application :

Tâches	1	2	3	4	5	6	7
Assemblage	20	12	19	16	14	12	17
Inspection	4	1	9	12	5	7	8
Expédition	7	11	4	18	18	3	6

Reformulation du problème :

•la 1ère machine regroupe les machines A et B

$t_{iAB} = t_{iA} + t_{iB}$

•la 2ème machine regroupe les machines B et C

$t_{iBC} = t_{iB} + t_{iC}$

Algorithme de Johnson

Application :

Tâches	1	2	3	4	5	6	7
Assemblage	20	12	19	16	14	12	17
Inspection	4	1	9	12	5	7	8
Expédition	7	11	4	18	18	3	6

Problème reformulé :

Tâches	1	2	3	4	5	6	7
Assemblage + Inspection	24	13	28	28	19	19	25
Inspection + Expédition	11	12	13	30	23	10	14

Algorithme de Johnson

Application :

Problème reformulé :

Tâches	1	2	3	4	5	6	7
Assemblage + Inspection	24	13	28	28	19	19	25
Inspection + Expédition	11	12	13	30	23	10	14

Application de l'algorithme de Johnson à deux machines :

Place	1	2	3	4	5	6	7
Tâche	5	4	7	3	2	1	6

Exercice

Un responsable d'entrepôt reçoit une commande de chargement pour 10 palettes bien spécifiques. Chaque palette doit être assemblée (atelier A), emballée (atelier B) puis inspectée et chargée (atelier C) dans le camion. Les temps en heures des différentes opérations sont donnés par le tableau suivant :

Palette i	1	2	3	4	5	6	7	8	9	10
Temps assemblage t_{iA}	1h10′	0h55′	0h45′	0h20′	1h15′	1h25′	0h55′	0h40′	1h05′	1h00′
Temps emballage t_{iB}	0h25′	0h30′	0h35′	0h25′	0h40′	0h35′	0h40′	0h25′	0h20′	0h30′
Temps inspection + chargement t_{iC}	0h45′	0h55′	1h10′	1h05′	0h50′	1h15′	0h45′	0h50′	1h05′	1h20′

Comme seulement 3 opérateurs sont disponibles pour réaliser ce travail, les palettes sont réalisées l'une après l'autre et sont traitées sur A, puis sur B, puis sur C.

1. Pour chaque atelier, appliquer la règle T.O.M afin de minimiser le temps d'achèvement moyen T_i tout en précisant cette valeur.
2. On considère ensuite les 3 ateliers simultanément.
(a) En justifiant des conditions nécessaires à l'utilisation de l'algorithme de Johnson, donner l'ordre de traitement des palettes permettant de minimiser le temps nécessaire à la réalisation de la commande.
(b) Tracer un diagramme de Gantt correspondant au problème. Préciser le temps minimal nécessaire à la réalisation de la commande.

Algorithme de CDS (Campbell, Dudek et Smith)

- Cas de **m** machines
- Ordre identique de passage
- $(n!)^m$ **ordonnancements possibles**
- **Algorithme CDS :** c'est l'algorithme de Johnson sur des groupements de machines successives

Exemple de problème de flow shop à 4 centres de production

	Temps d'exécution en 1/10ème d'heure			
Tâche *i*	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 3 problèmes suivants : {A} – {D};{AB} – {CD};{ABC} – {BCD}

Algorithme de CDS (Campbell, Dudek et Smith)

- Exemple :

Exemple de problème de flow shop à 4 centres de production

	Temps d'exécution en 1/10ème d'heure			
Tâche *i*	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

1^er problème fictif {A} – {D} :

Ordonnancement obtenu : 6 – 3 – 4 – 2 – 5 – 1

A_j = 51.2 heures

Algorithme de CDS (Campbell, Dudek et Smith)

- Exemple :

$2^{ème}$ problème fictif : {AB} – {CD}

Deuxième problème fictif de l'algorithme CDS

Tâche i	$t_{i,A+B}$	$t_{i,C+D}$
1	93	19
2	188	172
3	54	118
4	72	106
5	80	79
6	81	144

Ce deuxième problème ({AB}–{CD}) donne la solution suivante : 3 – 4 – 6 – 2 – 5 – 1
Avec cet ordonnancement, l'ensemble des travaux sera terminé au bout de 48,7 heures.

Algorithme de CDS (Campbell, Dudek et Smith)

- <u>Exemple :</u>

$$3^{\text{ème}} \text{ problème fictif } : \{ABC\} - \{BCD\}$$

Troisième problème fictif de l'algorithme CDS

Tâche i	$t_{i, A+B+C}$	$t_{i, B+C+D}$
1	108	62
2	283	271
3	74	165
4	84	170
5	145	98
6	147	224

Ce dernier problème ($\{ABC\}-\{BCD\}$) donne la même solution que le deuxième problème

Algorithme de CDS (Campbell, Dudek et Smith)

- Exemple :

Solution retenue :

Deuxième problème fictif de l'algorithme CDS

Tâche i	$t_{i,\ A+B}$	$t_{i,\ C+D}$
1	93	19
2	188	172
3	54	118
4	72	106
5	80	79
6	81	144

Ce deuxième problème ({AB}–{CD}) donne la solution suivante : 3 – 4 – 6 – 2 – 5 – 1
Avec cet ordonnancement, l'ensemble des travaux sera terminé au bout de 48,7 heures.

Utilisation des heuristiques :

- Algorithme de Johnson généralisé
- Heuristique NEH (Nawaz-Enscore-Ham)
- Heuristique de Palmer
- Heuristique de Hundal et Rajgopal :
- Heuristique de Chen
- Heuristique de Ham
- Heuristique de Gupta
- Heuristique de PHD (Procédure Han et Dejax)

Algorithme de Johnson généralisé

1. Calculer pour chaque produit :

- x = somme des n-1 premières tâches (dernière tâche exclue)
- y = somme des n-1 dernières tâches (première tâche exclue)
- k = x / y

2. L'ordonnancement est défini par l'ordre croissant du rapport k.

Algorithme de Johnson généralisé

Exemple :

	Temps d'exécution (min)						
Tâche i	t_{iA}	t_{iB}	t_{iC}	t_{iD}	**x**	**y**	**k = x/y**
1	4	2	3	5	9	10	0.90
2	6	5	4	6	15	15	1.00
3	2	1	8	1	11	10	1.10
4	8	7	2	3	17	12	1.42

Ordonnancement retenu : 1 - 2 - 3 - 4

Algorithme de Johnson généralisé

Exercice :

	Temps d'exécution (min)			
Tâche i	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

Heuristique NEH (Nawaz-Enscore-Ham)

- Trier les jobs par $\sum_{j=1}^{m}(P_{ij})$ croissant.
- Ordonnancer les deux premiers travaux dans l'ordre minimisant le C_{max}.
- Ensuite, prendre chaque autre travail dans la liste triée et l'insérer à la meilleure position pour la minimisation du C_{max} dans l'ordonnancement partiel.

Heuristique NEH (Nawaz-Enscore-Ham)

- Trier les jobs par $\sum_{j=1}^{m}(P_{ij})$ croissant.
- Ordonnancer les deux premiers travaux dans l'ordre minimisant le C_{max}.
- Ensuite, prendre chaque autre travail dans la liste triée et l'insérer à la meilleure position pour la minimisation du C_{max} dans l'ordonnancement partiel.

Exemple :

	M_1	M_2	M_3	
	P_{i1}	P_{i2}	P_{i3}	$\sum_{j=1}^{m} Pij$
J_1	2	5	1	8
J_2	4	3	2	9
J_3	6	1	4	11
J_4	4	2	2	8

Heuristique NEH (Nawaz-Enscore-Ham)

Exemple :

	M_1	M_2	M_3	
	P_{i1}	P_{i2}	P_{i3}	$\sum_{j=1}^{m} Pij$
J_1	2	5	1	8
J_2	4	3	2	9
J_3	6	1	4	11
J_4	4	2	2	8

– On trie les jobs par $\sum_{j=1}^{m}(P_{ij})$ croissant.

J_1	J_4	J_2	J_3

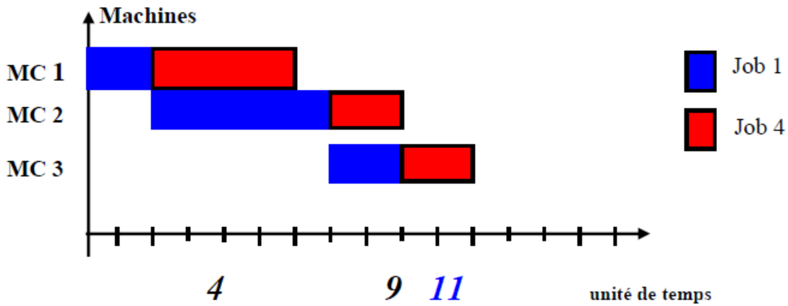
Heuristique NEH (Nawaz-Enscore-Ham)

Exemple :

	M_1	M_2	M_3	
	P_{i1}	P_{i2}	P_{i3}	$\sum_{j=1}^{m} Pij$
J_1	2	5	1	8
J_2	4	3	2	9
J_3	6	1	4	11
J_4	4	2	2	8

- On ordonnance les deux premiers travaux dans l'ordre minimisant le C_{max}

❖ (J_1, J_4) :



⇨ $C_{max} = 11$

Les Heuristiques

2. Ordonnancement d'atelier Flow shop

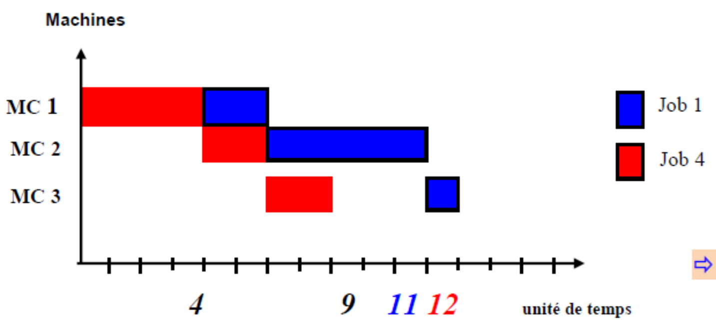
Heuristique NEH (Nawaz-Enscore-Ham)

Exemple :

	M_1	M_2	M_3	
	P_{i1}	P_{i2}	P_{i3}	$\sum_{j=1}^{m} Pij$
J_1	2	5	1	8
J_2	4	3	2	9
J_3	6	1	4	11
J_4	4	2	2	8

- On ordonnance les deux premiers travaux dans l'ordre minimisant le C_{max}

❖ (J_4, J_1) :



⇨ $C_{max} = 12$

⇨ $J_1 \rightarrow J_4$

Heuristique NEH (Nawaz-Enscore-Ham)

Exemple :

	M_1	M_2	M_3	
	P_{i1}	P_{i2}	P_{i3}	$\sum_{j=1}^{m} P_{ij}$
J_1	2	5	1	8
J_2	4	3	2	9
J_3	6	1	4	11
J_4	4	2	2	8

– Ensuite, on prend chaque autre travail dans la liste triée et l'insérer à la meilleure position pour la minimisation du C_{max} dans l'ordonnancement partiel.

Insérer J2

$(J_2,J_1,J_4) \Rightarrow C_{max} = 16$

$(J_1,J_2,J_4) \Rightarrow C_{max} = 14$

$(J_1,J_4,J_2) \Rightarrow C_{max} = 15$

Insérer J3

$(J_3,J_1,J_2,J_4) \Rightarrow C_{max} = 20$

$(J_1,J_3,J_2,J_4) \Rightarrow C_{max} = 20$

$(J_1,J_2,J_3,J_4) \Rightarrow C_{max} = 20$

$(J_1,J_2,J_4,J_3) \Rightarrow C_{max} = 21$

Heuristique NEH (Nawaz-Enscore-Ham)

Exercice :

	M_1	M_2	M_3
	P_{i1}	P_{i2}	P_{i3}
J_1	2	5	1
J_2	4	3	2
J_3	8	2	5
J_4	4	4	2
J_5	2	5	4

Heuristique de Palmer

Principe : donner une priorité aux tâches ayant des temps d'exécution croissants dans leur gamme opératoire. Cette heuristique contient deux étapes :

<u>Etape 1 :</u>

Pour chaque tâche (i=1,…,n), calculer la valeur de l'indice f(i) tel que :

$$f(i) = \sum_{j=1}^{m} (m - 2j + 1)\, t_{ij} \qquad \text{pour} \quad i = 1 \ldots n$$

<u>Etape 2 :</u>

La séquence est déterminée en classant les tâches par ordre croissant des indices f(i)

Heuristique de Palmer

Exemple :

$$f(i) = \sum_{j=1}^{m} (m - 2j + 1)\, t_{ij} \qquad \text{pour} \quad i = 1 \ldots n$$

Machines j / Tâches i	M_1	M_2	M_3	f(i)
	t_{i1}	t_{i2}	t_{i3}	
1	2	5	1	
2	4	3	2	
3	8	2	5	
4	4	4	2	

Heuristique de Palmer

Exemple :

$$f(i) = \sum_{j=1}^{m} (m - 2j + 1)\, t_{ij} \qquad \text{pour} \quad i = 1 \ldots n.$$

Machines j / Tâches i	M_1	M_2	M_3	f(i)
	t_{i1}	t_{i2}	t_{i3}	
1	2	5	1	2
2	4	3	2	4
3	8	2	5	6
4	4	4	2	4

Ordonnancement retenu : 1 - 2 - 4 - 3

Heuristique de Palmer

Exercice :

Tâches i \ Machines j	M_1	M_2	M_3
	t_{i1}	t_{i2}	t_{i3}
1	2	5	1
2	4	3	2
3	8	2	5

Heuristique de Hundal et Rajgopal

Etapes :

1. On établit un 1er ordonnancement, grâce à l'heuristique de Palmer

2. On bâtit une pseudo-machine avec les temps opératoires fictifs suivants :

$$t'_i = \sum_{j=1}^{m} (m - 2j)\, t_{ij}, \qquad \forall_i = 1 \ldots n$$

3. On classe les tâches dans l'ordre des t_i' décroissants et on obtient un 2ème ordonnancement

Heuristique de Hundal et Rajgopal

Etapes :

4. On bâtit une autre pseudo-machine avec :

$$t_i'' = \sum_{j=1}^{m} (m-2j+2)\, t_{ij}, \qquad \forall_i = 1 \ldots n$$

5. On classe les tâches dans l'ordre des t_i" décroissants et on obtient un 3ème ordonnancement

- L'ordonanncement qui, parmi les trois, conduit au plus petit C_{max} est retenu.

Heuristique de Chen

Etape 1 :

On calcule la somme des temps opératoires S(i) pour chaque tâche i

$$S(i) = \sum_{j=1}^{m} t_{ij}, \qquad \forall_i = 1 \ldots n$$

On trouve la tâche **c** qui a la S(i) la plus importante et on enlève cette tâche de l'ensemble de tâches non ordonnancées

Etape 2 :

- On ordonnance les tâches qui ont $t_{i1} \leq t_{im}$ dans l'ordre croissant de t_{i1} et on obtient un ordonnancement partiel S_A
- On ordonnance les tâches qui ont $t_{i1} > t_{im}$ dans l'ordre décroissant de t_{im} et on obtient un ordonnancement partiel S_B
- On retient comme solution finale l'ordonnancement (S_A, **c**, S_B)

Heuristique de Chen

Exemple :

Machines j / Tâches i	M_1	M_2	M_3	S(i)
	t_{i1}	t_{i2}	t_{i3}	
1	2	5	4	11
2	4	3	2	9
3	8	2	5	15
4	4	4	2	10

$S_A = (1)$

$S_B = (2, 4)$

Solution finale →

$S = (S_A, \mathbf{c}, S_B) = (1, 3, 2, 4)$

Heuristique de Chen

Exercice :

Machines j / Tâches i	M_1	M_2	M_3	M_4
	t_{i1}	t_{i2}	t_{i3}	t_{i4}
1	2	5	1	5
2	4	3	2	3
3	8	2	5	8
4	4	4	2	1
5	1	7	6	2

Heuristique de Ham

Etape 1 :

Décomposer la matrice initiale des temps opératoires en deux sous-matrices

• la première matrice regroupe les premières m/2 (si m est pair) ou (m+ l)/2 machines (sinon)

• la deuxième regroupe les dernières m/2 (si m est pair) ou (m + l)/2 machines (sinon)

Etape 2 :

On calcule la somme des temps opératoires pour chaque sous-matrice et pour chaque tâche i :

$$P_{i1} = \sum_{j=1}^{m/2} t_{ij} \quad \text{ou} \quad P_{i1} = \sum_{j=1}^{(m+1)/2} t_{ij},$$

$$P_{i2} = \sum_{j=(m+2)/2}^{m} t_{ij} \quad \text{ou} \quad P_{i2} = \sum_{j=(m+1)/2}^{m} t_{ij};$$

Heuristique de Ham

Etape 3 :

• On calcule $(P_{i2} - P_{i1})$ pour chaque tâche i , $i = 1, \ldots, n$

• On établit un 1er ordonnancement dans l'ordre décroissant de $(P_{i2} - P_{i1})$

• On établit un 2ème ordonnancement en classant les tâches qui ont $(P_{i2} - P_{i1})>0$ dans l'ordre croissant de P_{i1}

et les tâches qui ont $(P_{i2} - P_{i1})<0$ dans l'ordre décroissant de P_{i2}

• On retient celui parmi les deux qui conduit au plus petit C_{max} comme ordonnancement final.

Exemple :

Heuristique de Ham

Machines j / Tâches i	M_1	M_2	M_3	M_4	P_{i1}	P_{i2}	$P_{i2} - P_{i1}$
	t_{i1}	t_{i2}	t_{i3}	t_{i4}			
1	7	5	1	5	12	6	- 6
2	4	3	2	6	7	8	1
3	8	2	5	4	10	9	- 1
4	4	4	2	1	8	3	- 5

1er ordonnancement : 2 , 3 , 4 , 1 ($C_{max} = 34$)

2ème ordonnancement : $(P_{i2} - P_{i1}) > 0$ ordre ↗ P_{i1} ⇒ $\boxed{2 , 3 , 1 , 4}$ ($C_{max} = 32$)

$(P_{i2} - P_{i1}) < 0$ ordre ↘ P_{i2}

Heuristique de Ham

Exercice :

Machines j / Tâches i	M_1	M_2	M_3	M_4
	t_{i1}	t_{i2}	t_{i3}	t_{i4}
1	5	4	5	1
2	8	2	4	2
3	2	6	8	7
4	1	3	2	9

Heuristique de Gupta

Etape 1 :

Pour chaque tâche (i=1,…,n), calculer la valeur de l'indice f(i) tel que :

$$f(i) = \frac{\text{signe}(t_{i1} - t_{im})}{\min\limits_{1 \leq j \leq m-1} (t_{ij} + t_{ij+1})} \qquad \text{pour} \quad i = 1 \ldots n.$$

$$\text{signe}(t_{i1} - t_{im}) = \begin{cases} 1 & \text{si} \quad t_{i1} - t_{im} \geq 0 \\ -1 & \text{sinon} \end{cases}.$$

Etape 2 :

La séquence est déterminée en classant les tâches par ordre croissant des indices f(i)

En cas d'égalité, on préfère celle ayant la plus faible somme des temps opératoires.

Heuristique de Gupta

Exemple :

Machines j / Tâches i	M_1	M_2	M_3	f(i)
	t_{i1}	t_{i2}	t_{i3}	
1	2	5	4	- 0.14
2	4	3	2	0.14
3	8	2	5	0.10
4	4	4	2	0.12

Ordonnancement retenu : 1 - 3 - 4 - 2

Heuristique de Gupta

Exercice :

Machines j / Tâches i	M_1	M_2	M_3
	t_{i1}	t_{i2}	t_{i3}
1	2	5	1
2	4	3	2
3	8	2	5
4	4	4	2
5	2	5	4

Heuristique de PHD (Procédure Han et Dejax)

Principe :

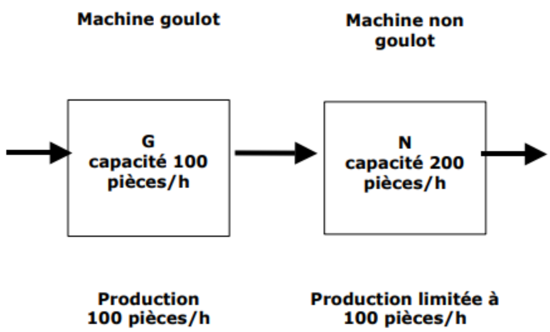
Problème de gestion des flux de tâches en présence d'une machine goulot.

→minimiser C_{max}
→faire passer le plus vite possible le flux dans le système avec respect des contraintes imposées, c'est la machine goulot.

Machine goulot : la machine la plus chargée qui détermine la vitesse du flux.

Il faut donc que la machine goulot commence à fonctionner le plus tôt possible et que les temps des travaux restant à faire sur les machines en aval soient minimisés une fois les travaux sur la machine goulot terminés.

Il faut également éviter l'arrêt de la machine goulot en cours de fonctionnement.



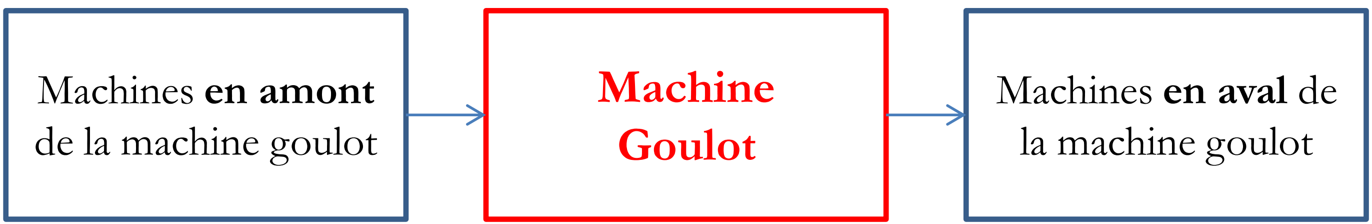
Heuristique de PHD (Procédure Han et Dejax)

<u>Principe :</u>

→Décomposition de l'ensemble des machines réelles en 2 machines fictives 1 et 2.

Dans le cas où la machine goulot est la première (respectivement la dernière), elle est considérée comme la première (respectivement la dernière) machine fictive.

1ère machine fictive 2ème machine fictive



Heuristique de PHD (Procédure Han et Dejax)

<u>Étapes de l'heuristique :</u>

Étape 1 : Calcul de la charge de chaque machine :

$$W_j = \sum_{i=1}^{n} t_{ij}, \qquad j = 1, \ldots, m$$

Soit **J** l'ensemble des machines j (j = 1, . . ., m)

Étape 2 : Choix d'une machine goulot :

Sélectionner la machine k, telle que :

$$\underset{k \in J}{W_k} = \underset{j \in J}{\mathrm{Max}}\, W_j$$

Heuristique de PHD (Procédure Han et Dejax)

<u>Étapes de l'heuristique :</u>

Étape 3 : Calcul des temps opératoires des 2 machines fictives :

Soit P_{i1} et P_{i2} les temps opératoires des tâches i, i = 1 , . . . , n , sur les machines fictives 1 et 2 :

Si k= 1 alors $P_{i1} = t_{i1}$

Si k=m alors $P_{i2} = t_{im}$

Sinon :

$$P_{i1} = \frac{1}{k-1} \sum_{j=1}^{k-1} t_{ij} \qquad P_{i2} = \frac{1}{m-k} \sum_{j=k+1}^{m} t_{ij}$$

Heuristique de PHD (Procédure Han et Dejax)

Étapes de l'heuristique :

Étape 4 : Calcul de la fonction de classement par :

$$f(i) = \frac{\text{signe}(P_{i1} - P_{i2})}{\text{Min}(P_{i1}, P_{i2})}$$

Avec :

$$\text{signe}(P_{i1} - P_{i2}) = \begin{cases} 1 & \text{si} \quad P_{i1} - P_{i2} \geqq 0 \\ -1 & \text{sinon} \end{cases}$$

Heuristique de PHD (Procédure Han et Dejax)

Étapes de l'heuristique :

Étape 5 : Détermination de l'ordonnancement.
Il correspond au classement des tâches en ordre croissant en fonction de $f(i)$, i= 1, . . .,n

La date d'achèvement est C_{max}^{k}

Étape 6 : Mise à jour.
Enlever k de l'ensemble J.

Si J = Ø , aller à l'étape 7
Sinon, revenir à 2

Heuristique de PHD (Procédure Han et Dejax)

Étapes de l'heuristique :

Étape 7 : Choix du meilleur ordonnancement.

Retenir comme solution l'ordonnancement correspondant à la machine goulot k telle que :

$$C^k_{\max} \leq C^j_{\max}, \qquad j = 1, \ldots, m$$

Heuristique de PHD (Procédure Han et Dejax)

Application :

Machines j / Tâches i	M_1	M_2	M_3
	t_{i1}	t_{i2}	t_{i3}
1	2	5	1
2	4	3	2
3	8	2	5

Heuristique de PHD (Procédure Han et Dejax)

Application :

Étape 1 : Calcul de la charge de chaque machine :

$$W_j = \sum_{i=1}^{n} t_{ij}, \qquad j = 1, \ldots, m$$

Soit **J** l'ensemble des machines j (j = 1, . . ., m)

Heuristique de PHD (Procédure Han et Dejax)

Application :

Machines j / Tâches i	M_1	M_2	M_3
	t_{i1}	t_{i2}	t_{i3}
1	2	5	1
2	4	3	2
3	8	2	5
W_j	**14**	**10**	**8**

Heuristique de PHD (Procédure Han et Dejax)

Application :

Étape 2 : Choix d'une machine goulot,

Sélectionner la machine k, telle que :

$$\underset{k \in J}{W_k} = \underset{j \in J}{\text{Max}}\, W_j$$

Heuristique de PHD (Procédure Han et Dejax)

Application :

Machines j / Tâches i	M_1	M_2	M_3
	t_{i1}	t_{i2}	t_{i3}
1	2	5	1
2	4	3	2
3	8	2	5
W_j	**14**	**10**	**8**

Heuristique de PHD (Procédure Han et Dejax)

Application :

Étape 3 : Calcul des temps opératoires des 2 machines fictives :

Soit P_{i1} et P_{i2} les temps opératoires des tâches i, i = 1 , . . . , n , sur les machines fictives 1 et 2 :

Si k= 1 alors $P_{i1} = t_{i1}$

Si k=m alors $P_{i2} = t_{im}$

Sinon :

$$P_{i1} = \frac{1}{k-1} \sum_{j=1}^{k-1} t_{ij}$$

$$\boxed{P_{i2} = \frac{1}{m-k} \sum_{j=k+1}^{m} t_{ij}}$$

Heuristique de PHD (Procédure Han et Dejax)

Application :

Machines j / Tâches i	M_1	M_2	M_3	P_{i1}	P_{i2}
	t_{i1}	t_{i2}	t_{i3}		
1	2	5	1	2	3
2	4	3	2	4	2.5
3	8	2	5	8	3.5
W_j	**14**	**10**	**8**		

Si k= 1 alors $P_{i1} = t_{i1}$

$$P_{i2} = \frac{1}{m-k} \sum_{j=k+1}^{m} t_{ij}$$

Heuristique de PHD (Procédure Han et Dejax)

Application :

Étape 4 : Calcul de la fonction de classement par :

$$f(i) = \frac{\text{signe}\,(P_{i1} - P_{i2})}{\text{Min}\,(P_{i1}, P_{i2})}$$

Avec :

$$\text{signe}\,(P_{i1} - P_{i2}) = \begin{cases} 1 & \text{si} \quad P_{i1} - P_{i2} \geqq 0 \\ -1 & \text{sinon} \end{cases}$$

Les Heuristiques

2. Ordonnancement d'atelier Flow shop

Heuristique de PHD (Procédure Han et Dejax)

Application :

Machines j / Tâches i	M_1	M_2	M_3	P_{i1}	P_{i2}	f(i)
	t_{i1}	t_{i2}	t_{i3}			
1	2	5	1	2	3	- 0.5
2	4	3	2	4	2.5	0.4
3	8	2	5	8	3.5	0.28
W_j	**14**	**10**	**8**			

Heuristique de PHD (Procédure Han et Dejax)

Application :

Étape 5 : Détermination de l'ordonnancement.

Il correspond au classement des tâches en ordre croissant en fonction de $f(i)$, $i= 1, . . ., n$

La date d'achèvement est C_{max}^{k}

Heuristique de PHD (Procédure Han et Dejax)

Application :

Machines j / Tâches i	M_1	M_2	M_3	P_{i1}	P_{i2}	f(i)
	t_{i1}	t_{i2}	t_{i3}			
1	2	5	1	2	3	- 0.5
2	4	3	2	4	2.5	0.4
3	8	2	5	8	3.5	0.28
W_j	**14**	**10**	**8**			

1er ordonnancement : 1 – 3 – 2

$C_{max}^{1} = 19$

Heuristique de PHD (Procédure Han et Dejax)

Application :

Étape 6 : Mise à jour.

Enlever k de l'ensemble J.

Si $J = \emptyset$, aller à l'étape 7

Sinon, revenir à 2

Heuristique de PHD (Procédure Han et Dejax)

Application :

Étape 6 : Mise à jour.
Enlever k de l'ensemble J.

Si J = Ø , aller à l'étape 7
Sinon, revenir à 2

Machines j / Tâches i		M_2	M_3
		t_{i2}	t_{i3}
1		5	1
2		3	2
3		2	5
W_j		**10**	**8**

Heuristique de PHD (Procédure Han et Dejax)

Application :

Étape 2 : Choix d'une machine goulot,

Sélectionner la machine k, telle que :

$$\underset{k \in J}{W_k} = \underset{j \in J}{\mathrm{Max}}\, W_j$$

Heuristique de PHD (Procédure Han et Dejax)

Application :

Machines j / Tâches i		M_2	M_3
		t_{i2}	t_{i3}
1		5	1
2		3	2
3		2	5
W_j		**10**	**8**

Heuristique de PHD (Procédure Han et Dejax)

Application :

Étape 3 : Calcul des temps opératoires des 2 machines fictives :

Soit P_{i1} et P_{i2} les temps opératoires des tâches i, i = l , . . . , n , sur les machines fictives 1 et 2 :

Si k= 1 alors $P_{i1} = t_{i1}$

Si k=m alors $P_{i2} = t_{im}$

Sinon :

$$P_{i1} = \frac{1}{k-1} \sum_{j=1}^{k-1} t_{ij}$$

$$\boxed{P_{i2} = \frac{1}{m-k} \sum_{j=k+1}^{m} t_{ij}}$$

Heuristique de PHD (Procédure Han et Dejax)

Application :

Machines j / Tâches i		M_2 t_{i2}	M_3 t_{i3}	P_{i1}	P_{i2}
1		5	1	5	1
2		3	2	3	2
3		2	5	2	5
W_j		**10**	**8**		

Si k= 1 alors $P_{i1} = t_{i1}$

$$P_{i2} = \frac{1}{m-k} \sum_{j=k+1}^{m} t_{ij}$$

Heuristique de PHD (Procédure Han et Dejax)

Application :

Étape 4 : Calcul de la fonction de classement par :

$$f(i) = \frac{\text{signe}(P_{i1} - P_{i2})}{\text{Min}(P_{i1}, P_{i2})}$$

Avec :

$$\text{signe}(P_{i1} - P_{i2}) = \begin{cases} 1 & \text{si} \quad P_{i1} - P_{i2} \geqq 0 \\ -1 & \text{sinon} \end{cases}$$

Heuristique de PHD (Procédure Han et Dejax)

Application :

Machines j / Tâches i		M_2	M_3	P_{i1}	P_{i2}	f(i)
		t_{i2}	t_{i3}			
1		5	1	5	1	1
2		3	2	3	2	0.5
3		2	5	2	5	- 0.5
W_j		**10**	**8**			

$$f(i) = \frac{\text{signe}(P_{i1} - P_{i2})}{\text{Min}(P_{i1}, P_{i2})}$$

$$\text{signe}(P_{i1} - P_{i2}) = \begin{cases} 1 & \text{si} \quad P_{i1} - P_{i2} \geqq 0 \\ -1 & \text{sinon} \end{cases}$$

Heuristique de PHD (Procédure Han et Dejax)

Application :

Étape 5 : Détermination de l'ordonnancement.
Il correspond au classement des tâches en ordre croissant en fonction de $f(i)$, $i = 1, \ldots, n$

La date d'achèvement est C_{max}^{k}

Heuristique de PHD (Procédure Han et Dejax)

Application :

Machines j / Tâches i		M_2 / t_{i2}	M_3 / t_{i3}	P_{i1}	P_{i2}	f(i)
1		5	1	5	1	1
2		3	2	3	2	0.5
3		2	5	2	5	- 0.5
W_j		**10**	**8**			

2ème ordonnancement : 3 – 2 – 1

$C_{max}^2 = 21$

Heuristique de PHD (Procédure Han et Dejax)

Application :

Étape 6 : Mise à jour.
Enlever k de l'ensemble J.

Si $J = \emptyset$, aller à l'étape 7
Sinon, revenir à 2

Heuristique de PHD (Procédure Han et Dejax)

Application :

Étape 7 : Choix du meilleur ordonnancement.
Retenir comme solution l'ordonnancement correspondant à la machine goulot k telle que :

$$C_{\max}^{k} \leqq C_{\max}^{j}, \qquad j = 1, \ldots, m$$

1er ordonnancement : 1 – 3 – 2
C_{max}^{1} = 19

2ème ordonnancement : 3 – 2 – 1
C_{max}^{2} = 21

Ordonnancement final :
1 – 3 – 2

Heuristique de PHD (Procédure Han et Dejax)

Exercice :

Machines j / Tâches i	M_1	M_2	M_3
	t_{i1}	t_{i2}	t_{i3}
1	2	5	1
2	4	3	2
3	8	2	5
4	1	6	4

1. Méthodes de résolution approchées

Les Heuristiques

Les heuristiques sont des méthodes empiriques qui donnent généralement de bons résultats sans pour autant être démontrables. Elles se basent sur des règles simplifiées pour optimiser un ou plusieurs critères. Le principe général de cette catégorie de méthodes est d'intégrer des stratégies de décision pour construire une solution proche de celle optimale tout en cherchant à avoir un temps de calcul raisonnable.

- Ordonnancement d'atelier à machine unique
- Ordonnancement d'atelier Flow shop
- **Ordonnancement d'atelier Job shop**
- Ordonnancement d'atelier Open shop

Heuristique de Jackson

- Il constitue une extension de celui de Johnson.
- Il traite de plus le cas où certains jobs peuvent n'avoir à passer que sur une des deux machines.

- **Etape 1 : Partition des jobs en 4 ensembles :**
  - {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

- **Etape 2 : Ordonnancement des 4 ensembles :**
  - Algorithme de Johnson sur {AB}
  - Algorithme de Johnson sur {BA}
  - Ordre quelconque sur {A}
  - Ordre quelconque sur {B}

- **Etape 3 : Ordonnancement optimal :**
  - A : {AB} --> {A} --> {BA}
  - B : {BA} --> {B} --> {AB}

Heuristique de Jackson

Exemple :

J_i	J_1		J_2		J_3		J_4		J_5	
O_{ij}	O_{11}	O_{21}	O_{12}	O_{22}	O_{13}	O_{23}	O_{14}	O_{24}	O_{15}	O_{25}
M_k	M_1	M_2	M_1	--	M_1	M_2	M_2	M_1	--	M_2
P_{ij}	3	4	7	--	6	3	4	5	--	4

J_i	J_6		J_7		J_8		J_9	
O_{ij}	O_{16}	O_{26}	O_{17}	O_{27}	O_{18}	O_{28}	O_{19}	O_{29}
M_k	M_1	M_2	M_2	M_1	M_1	M_2	M_2	M_1
P_{ij}	5	4	4	2	2	4	6	5

Heuristique de Jackson

Exemple :

Ensemble	Jobs	Ordonnancement
{1}	{ J_2 }	{ J_2 }
{2}	{ J_5 }	{ J_5 }
{12}	{ J_1 , J_3 , J_6 , J_8 }	{ J_8 , J_1 , J_6 , J_3 }
{21}	{ J_4 , J_7 , J_9 }	{ J_4 , J_9 , J_7 }

Ordonnancement final :

Machine M_1 : {12} --> {1} --> {21} : { J_8 , J_1 , J_6 , J_3 , J_2 , J_4 , J_9 , J_7 }
Machine M_2 : {21} --> {2} --> {12} : { J_4 , J_9 , J_7 , J_5 , J_8 , J_1 , J_6 , J_3 }

Heuristique de Jackson

Exercice :

J_i	J_1		J_2		J_3		J_4		J_5	
O_{ij}	O_{11}	O_{21}	O_{12}	O_{22}	O_{13}	O_{23}	O_{14}	O_{24}	O_{15}	O_{25}
M_k	M_2	M_1	M_1	M_2	M_2	M_1	--	M_2	M_2	M_1
P_{ij}	4	5	7	8	6	2	--	5	8	4

J_i	J_6		J_7		J_8		J_9		J_{10}	
O_{ij}	O_{16}	O_{26}	O_{17}	O_{27}	O_{18}	O_{28}	O_{19}	O_{29}	O_{110}	O_{210}
M_k	--	M_2	M_1	--	M_1	--	M_2	M_1	M_2	M_1
P_{ij}	--	4	4	--	2	--	6	4	8	5

III. Méthodes de résolution des problèmes d'ordonnancement

1. Méthodes de résolution approchées

Les Heuristiques

Les heuristiques sont des méthodes empiriques qui donnent généralement de bons résultats sans pour autant être démontrables. Elles se basent sur des règles simplifiées pour optimiser un ou plusieurs critères. Le principe général de cette catégorie de méthodes est d'intégrer des stratégies de décision pour construire une solution proche de celle optimale tout en cherchant à avoir un temps de calcul raisonnable.

- Ordonnancement d'atelier à machine unique
- Ordonnancement d'atelier Flow shop
- Ordonnancement d'atelier Job shop
- **Ordonnancement d'atelier Open shop**

Règle LAPT

(Longest Alternate Processing Time first)

→Minimiser la durée d'achèvement de l'ensemble des tâches A_j

Sélectionner, sur le centre de production libre, l'opération de plus grande durée sur l'autre centre. Dans l'application de cette règle, les tâches dont une opération a été réalisée ont la même priorité, la plus faible, et, lorsque ces dernières sont considérées, elles le sont de manière arbitraire. Cette règle reste optimale dans le cas préemptif.

Exemple de problème d'ordonnancement à cheminement libre dans 2 centres de production

Numéro de la tâche *i*	1	2	3	4
t_{iA}	50	150	80	200
t_{iB}	60	50	100	70

Les Heuristiques

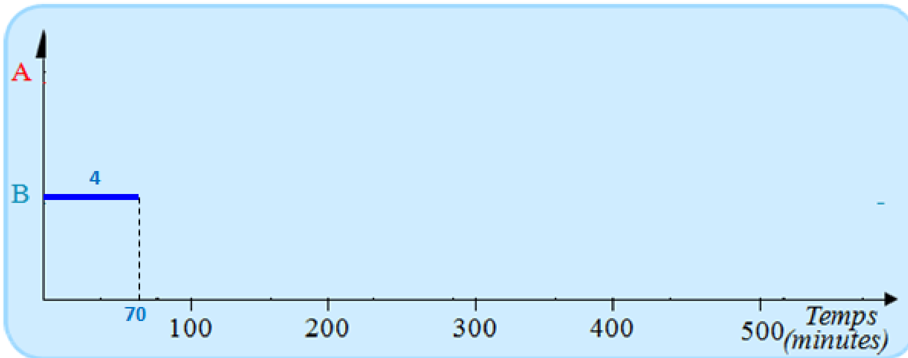
4. Ordonnancement d'atelier Open shop (cheminement libre)

Règle LAPT

Exemple de problème d'ordonnancement à cheminement libre dans 2 centres de production

Numéro de la tâche *i*	1	2	3	4
t_{iA}	50	150	80	200
t_{iB}	60	50	100	70

- en t = 0, l'opération la plus longue est l'opération A de la tâche 4, on charge donc l'opération de la tâche 4 à exécuter sur B (date d'achèvement en t = 70) (choisie arbitrairement parmi les 2 centres libres);

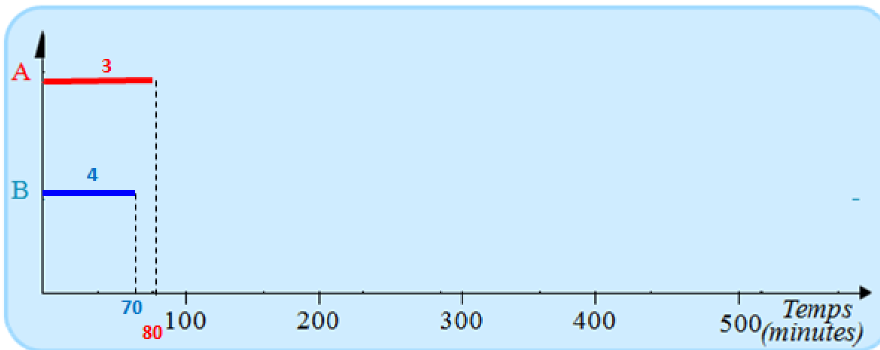


Règle LAPT

Exemple de problème d'ordonnancement à cheminement libre dans 2 centres de production

Numéro de la tâche i	1	2	3	4
t_{iA}	50	150	80	200
t_{iB}	60	50	100	70

- en t = 0, le centre A étant libre, les opérations des tâches 1, 2 et 3 sont donc candidates; l'opération la plus longue sur le centre B parmi ces candidats est celle de la tâche 3 (100), ce qui conduit à charger l'opération A de la tâche 3 (date d'achèvement en t = 80);

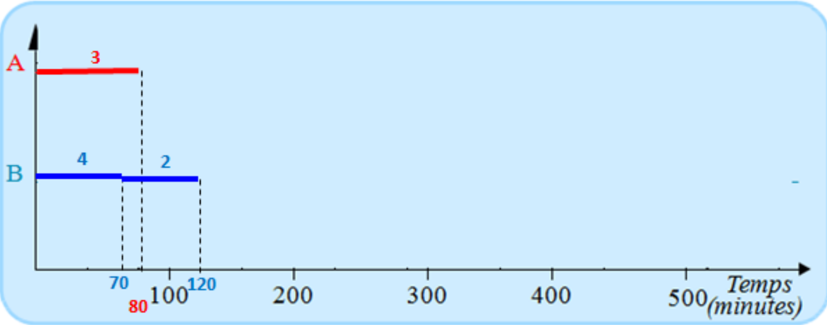


Règle LAPT

Exemple de problème d'ordonnancement à cheminement libre dans 2 centres de production

Numéro de la tâche i	1	2	3	4
t_{iA}	50	150	80	200
t_{iB}	60	50	100	70

-en t = 70, le centre B se libère, les opérations des tâches 1 et 2 sont donc candidates (3 étant en cours); l'opération la plus longue sur A de ces candidats est celle de la tâche 2 (150); on charge donc l'opération B de la tâche 2 (date d'achèvement en t = 70 + 50 = 120);

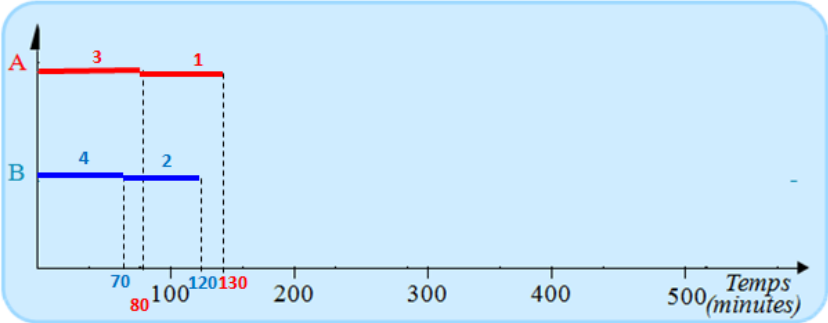


Règle LAPT

Exemple de problème d'ordonnancement à cheminement libre dans 2 centres de production

Numéro de la tâche *i*	1	2	3	4
t_{iA}	50	150	80	200
t_{iB}	60	50	100	70

en t = 80, le centre A se libère, les opérations des tâches 1 et 4 sont donc candidates; l'opération de la tâche 1 (qui n'a encore aucune réalisation de tâche) est plus prioritaire que celle de la tâche 4 (qui a déjà une opération réalisée); on charge donc l'opération A de la tâche 1 (date d'achèvement en t = 80+50 = 130);

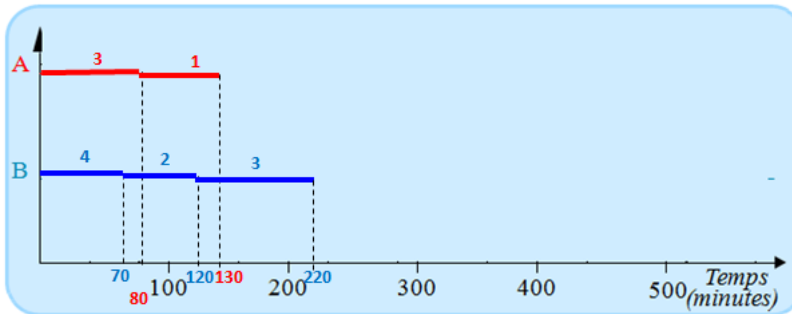


Règle LAPT

Exemple de problème d'ordonnancement à cheminement libre dans 2 centres de production

Numéro de la tâche *i*	1	2	3	4
t_{iA}	50	150	80	200
t_{iB}	60	50	100	70

- en t = 120, le centre B se libère, l'opération de la tâche 3 est candidate unique (1 étant en cours); on charge donc l'opération B de la tâche 3 (date d'achèvement en t = 120 + 100 = 220);

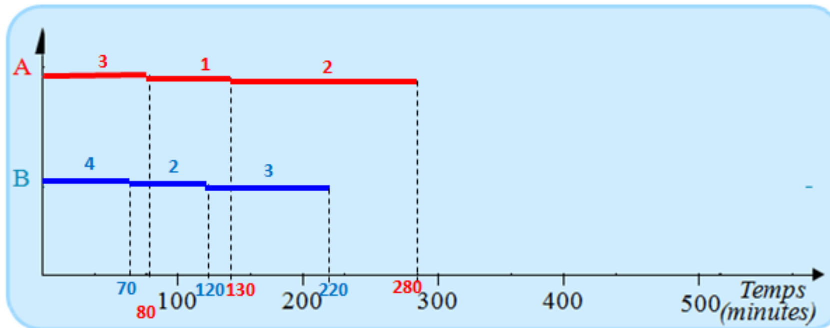


Règle LAPT

Exemple de problème d'ordonnancement à cheminement libre dans 2 centres de production

Numéro de la tâche *i*	1	2	3	4
t_{iA}	50	150	80	200
t_{iB}	60	50	100	70

- en t = 130, le centre A se libère, sont donc candidates les opérations des tâches 2 et 4 qui ont déjà toutes deux une opération exécutée sur B; elles sont équivalentes et on charge arbitrairement l'opération A de la tâche 2 (date d'achèvement en t = 130 + 150 = 280), puis l'opération A de la tâche 4 (date d'achèvement en t = 280 + 200 = 480) qui est la dernière opération à exécuter sur ce centre;

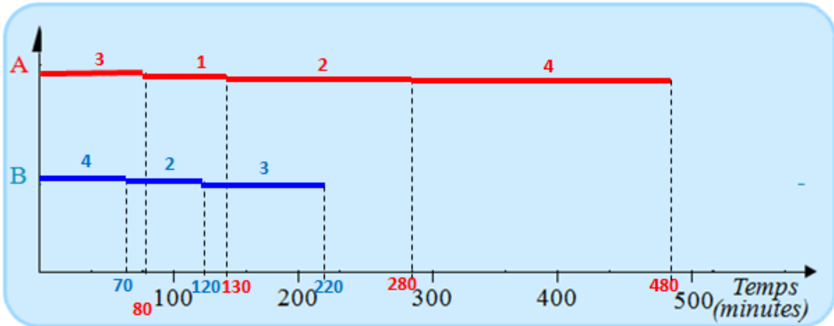


Règle LAPT

Exemple de problème d'ordonnancement à cheminement libre dans 2 centres de production

Numéro de la tâche i	1	2	3	4
t_{iA}	50	150	80	200
t_{iB}	60	50	100	70

- en t = 130, le centre A se libère, sont donc candidates les opérations des tâches 2 et 4 qui ont déjà toutes deux une opération exécutée sur B; elles sont équivalentes et on charge arbitrairement l'opération A de la tâche 2 (date d'achèvement en t = 130 + 150 = 280), puis l'opération A de la tâche 4 (date d'achèvement en t = 280 + 200 = 480) qui est la dernière opération à exécuter sur ce centre;

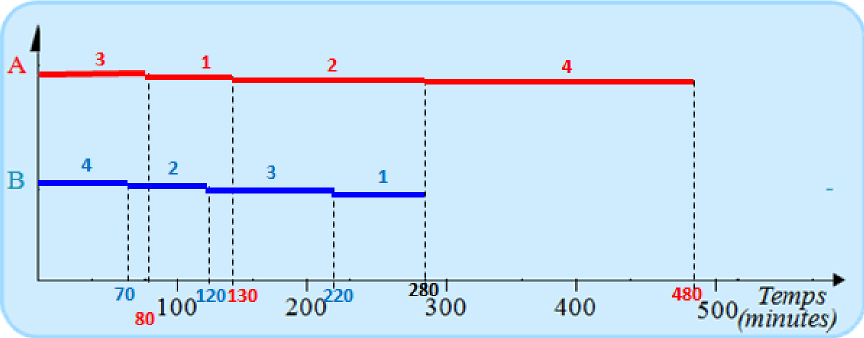


Règle LAPT

Exemple de problème d'ordonnancement à cheminement libre dans 2 centres de production

Numéro de la tâche *i*	1	2	3	4
t_{iA}	50	150	80	200
t_{iB}	60	50	100	70

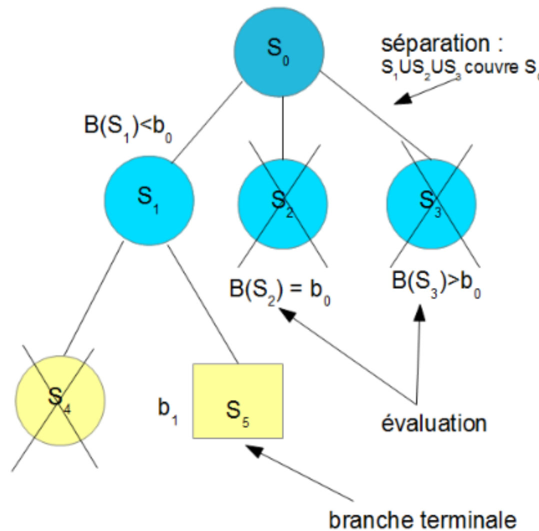
- en t = 220, le centre B se libère, on charge donc l'opération B de la tâche 1 (date d'achèvement en t=220+60 = 280) qui est la dernière à réaliser.



Introduction :

→Construire un arbre de recherche implicite. Cet arbre est exploré de façon à éviter les branches inutiles/dominées.

Branche dominée : partie de l'espace de recherche qui est vide des solutions réalisables ou bien ne contenant aucune solution plus intéressante qu'une solution courante. Cette exploration intelligente est réalisée grâce à des évaluations des branches et à des comparaisons avec un seuil de la valeur du critère (également appelé borne).



Principe :

La méthode se base sur les éléments suivants :

- la construction d'une solution heuristique,
- l'évaluation,
- la séparation
- l'exploration.

Principe :

1. **Construction d'une solution heuristique :**

La bonne d'une telle solution conditionne souvent le succès de cette méthode.

En effet, avec une solution initiale de bonne qualité, il est plus facile d'augmenter l'efficacité de l'exploration et de diminuer le temps de calcul. Néanmoins, la méthode peut théoriquement fonctionner avec toute solution heuristique réalisable pour le problème étudié.

Principe :

2. Séparation :

Elle consiste à décomposer l'univers des solutions en plusieurs sous-ensembles généralement disjoints.

Pour que la séparation soit valide, il est nécessaire que l'union de ces sous-ensembles couvre toutes les solutions possibles de l'ensemble séparé (ou bien de la branche séparée). Le principe de la séparation est récursif, conduisant ainsi à un arbre de recherche dont l'évolution, pendant le déroulement de l'algorithme, est liée aux sous-ensembles des solutions explorés.

Principe :

3. Evaluation :

L'évaluation consiste à associer une borne B au problème étudié que l'on calcule pour chaque branche explorée. On parle d'une borne inférieure (respectivement d'une borne supérieure) dans le cas d'une minimisation (respectivement dans le cas d'une maximisation).

Cette borne permet d'estimer la performance de la branche évaluée dans le meilleur des cas.

Principe :

4. Exploration :

L'exploration consiste à fixer un protocole donnant l'ordre de visite des différentes branches.

On peut choisir par exemple de commencer par explorer les branches les plus prometteuses (celles qui ont la valeur de la borne inférieure la plus petite dans le cas d'une minimisation) en estimant avoir plus de chance de trouver une bonne solution dans cette branche que d'en trouver dans les autres branches.

Principe :

4. Exploration :

Au cours de l'exploration et dans un problème de minimisation par exemple, on peut distinguer les cas suivants :

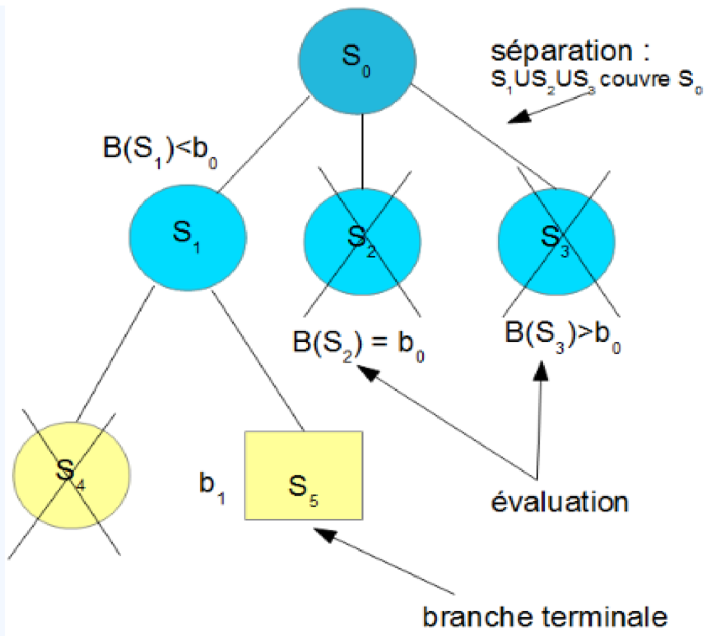
- la branche courante Si est évaluée par $B(Si) \geq b_0$, avec b_0 la borne supérieure initiale. Dans ce cas, il est inutile d'explorer la branche car dans le meilleur des cas, on va trouver une solution équivalente à la solution heuristique.
- la branche courante Si est évaluée par $B(Si) < b_0$, dans ce cas, la branche sera séparée et ensuite explorée.
- la branche est terminale : c'est à dire, elle correspond à une solution réalisable et par la suite, elle ne peut être séparée. Dans ce cas, si cette solution représente une valeur du critère $b_1 \geq b_0$, alors elle sera rejetée.

Si $b_1 < b_0$, la solution sera temporairement mémorisée comme étant la meilleure solution et elle remplacera la solution heuristique initiale. Dans ce cas, la valeur de b_0 sera remplacée par b_1.

Méthodes exactes

Séparation et Evaluation progressive (Branch and Bound)

Principe :



Exemple d'application :

tâche	A	B	C	D	E
r_i	0	4	8	7	11
p_i	4	2	1	3	4

$U=\{A,B,C,D,E\}$

r_i : date de début au plus tôt
p_i : temps d'exécution

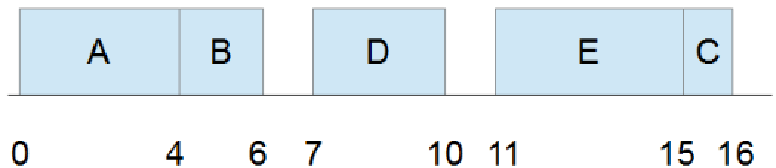
Objectif : minimiser C_{max}

La machine est disponible à $t_{disp}=0$

<u>Exemple d'application :</u>

tâche	A	B	C	D	E
r_i	0	4	8	7	11
p_i	4	2	1	3	4

Solution heuristique :



$b_0=16$

Exemple d'application :

tâche	A	B	C	D	E
r_i	0	4	8	7	11
p_i	4	2	1	3	4

Borne inférieure :

Basée sur la contrainte de disponibilité des tâches.

→Pour exécuter un sous ensemble V ⊂U , le temps nécessaire est minoré par la valeur B(V) définie comme suit :

$$B(V) = max\{t_{disp}, min_{i\in V}\{r_i\}\} + \sum_{i\in V} p_i$$

→ évaluer la performance de chaque branche séparée

Méthodes exactes

Séparation et Evaluation progressive (Branch and Bound)

Exemple d'application :

tâche	A	B	C	D	E
r_i	0	4	8	7	11
p_i	4	2	1	3	4

Étape initiale : évaluer la borne inférieure pour l'ensemble U (car on n'a pas encore séparé l'ensemble des solutions possibles).

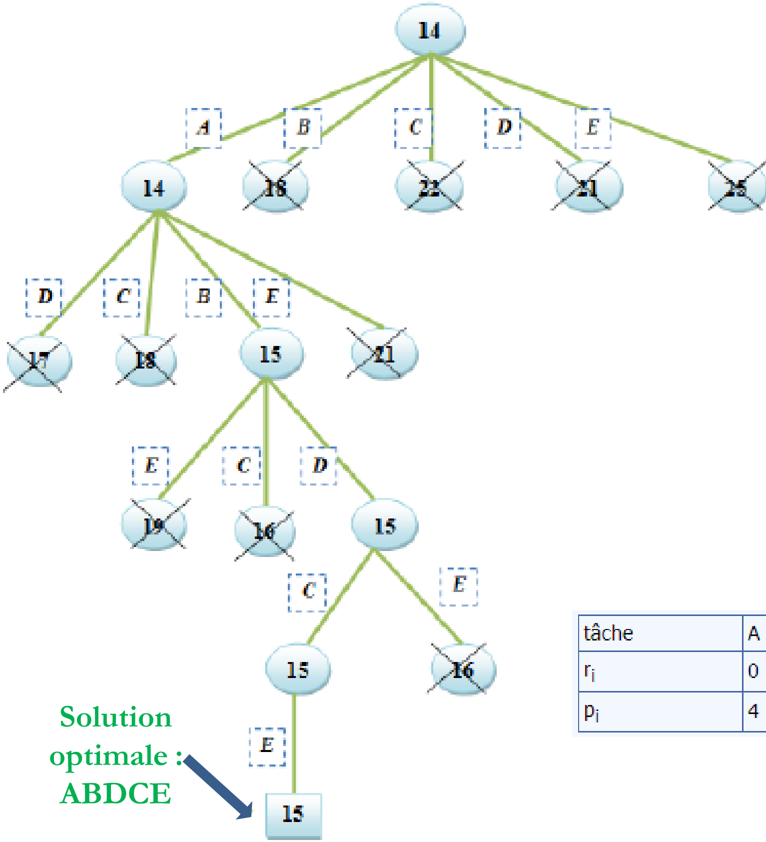
$$B(V) = max\{t_{disp}, min_{i\in V}\{r_i\}\} + \sum_{i\in V} p_i \qquad B(U)=14$$

→ évaluation de la branche primaire

Par la suite, la solution optimale a une valeur de critère comprise entre 14 et 16 (b_0=16)

Méthodes exactes

Séparation et Evaluation progressive (Branch and Bound)



$$B(V) = max\{t_{disp}, min_{i \in V}\{r_i\}\} + \sum_{i \in V} p_i$$

$b_0 = 16$

$B(U) = 14$

tâche	A	B	C	D	E
r_i	0	4	8	7	11
p_i	4	2	1	3	4