Introduction à l'ordonnancement de tâches

Christian Artigues

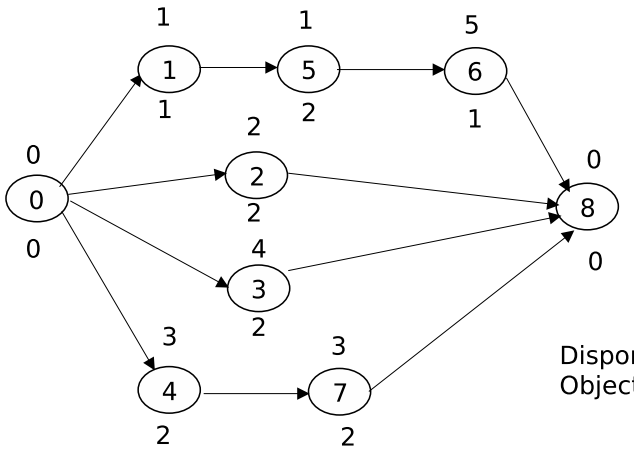
LAAS-CNRS

20 avril 2009

1. Introduction
2. Le problème central de l'ordonnancement
3. Classification des problèmes d'ordonnancement
4. Fonctions objectif régulières et ordonnancements dominants
5. Problèmes à une machine
6. Problèmes à machines parallèles
7. Problèmes d'atelier
8. Problèmes d'ordonnancement de projet avec contraintes de ressources

Définition informelle

Ordonnancer c'est organiser dans le temps l'exécution d'un ensemble de tâches soumises à des contraintes de temps et de ressources afin d'optimiser un objectif.



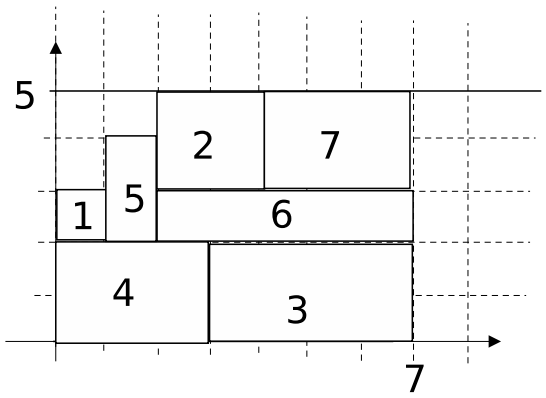
duree(i)

i

quantité requise(i)

Disponibilité de la ressource = 5
Objectif = minimiser la durée totale

Solution optimale



Types de problèmes d'ordonnancement

Ordonnancement statique/dynamique

Lorsque les données d'un problème d'ordonnancement sont connues à l'avance on parle de problème statique. Si les données apparaissent au cours du temps on parle de problème dynamique.

Ordonnancement déterministe/stochastique

Si à partir du moment où la donnée est connue, elle l'est avec certitude on parle de problème déterministe, sinon on parle de problème stochastique et on peut modéliser les données par des variables aléatoires ou tout autre représentation de l'incertain (intervalles, ...).

Ordonnancement acyclique/cyclique

Lorsque les tâches à ordonnancer doivent être répétées indéfiniment on parle de problème cyclique. Sinon, on parle de problème acyclique.

Ordonnancement acyclique, statique et déterministe

Un problème d'ordonnancement acyclique, statique et déterministe est un problème d'optimisation combinatoire.

Optimisation combinatoire

Etant donnés :

- Un ensemble discret N,
- une fonction d'ensemble $f : 2^N \rightarrow \mathbb{R}$,
- un ensemble $R \subseteq N$ de sous ensembles de N appelés solutions réalisables,

un problème d'optimisation combinatoire consiste à déterminer :

$$\min_{S \subseteq N} \{f(S) : S \in R\}$$

Comment résoudre un problème d'ordonnancement

Schéma classique de résolution des problèmes d'optimisation combinatoire

Détermination de la complexité

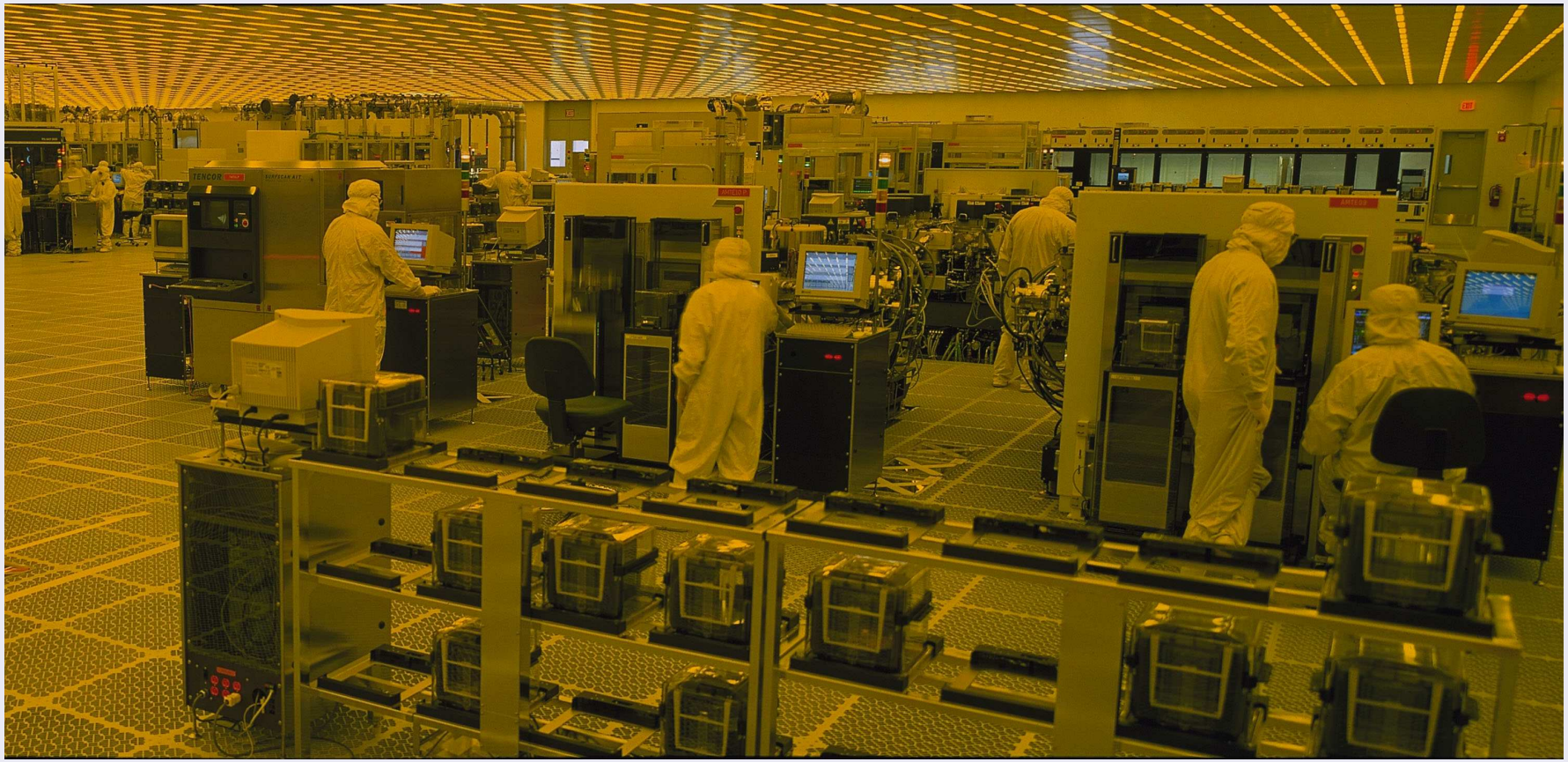
- Trouver un algorithme polynomial en fonction de la taille de l'entrée, ou bien
- Montrer que le problème est NP-difficile

Résolution d'un problème NP-difficile

- Si NP-difficile au sens faible, il existe un algorithme pseudo-polynomial (polynomial pour un codage unaire de l'entrée)
- Si NP-difficile au sens fort ou si algorithme pseudopolynomial non utilisable en pratique :
  - Méthodes exactes (programmation linéaire en nombres entiers, programmation dynamique, procédure de séparation et d'évaluation) ou approchées (heuristiques, metaheuristiques).

Applications industrielles 1/4

Ordonnancement de la production



Applications industrielles 2/4

Ordonnancement des services

Applications industrielles 3/4

Ordonnancement de projet

Applications industrielles 4/4

Ordonnancement de tâches informatiques

Définition du problème central de l'ordonnancement 1/3

Données du problème

Soit un ensemble de tâches $T = \{1, \ldots, n\}$, chaque tâche $i \in T$ étant caractérisée par

- sa durée $p_i \in \mathbb{R}$,
- sa date de lancement $r_i \in \mathbb{R}$,
- sa date d'échéance $\tilde{d}_i \in \mathbb{R}$,
- un ensemble de successeurs $\sigma(i)$ et pour chaque successeur $j \in \sigma(i)$, un délai $l_{ij} \in \mathbb{R}$.

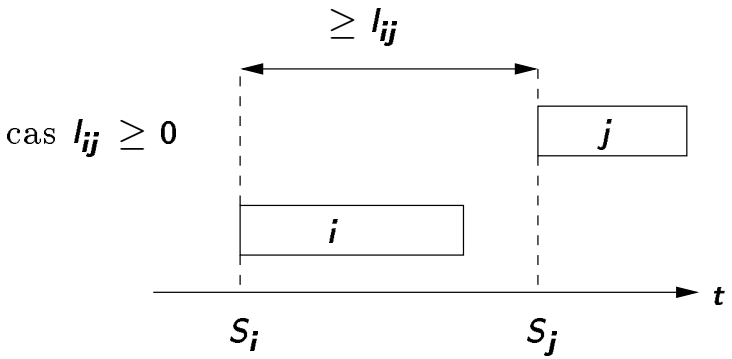
i	p_i	r_i	$\tilde{d}_i$	$\sigma(i)$	$l_{ij}, j \in \sigma(i)$
1	6	0	$+\infty$	2	1
2	4	0	$+\infty$	1	-3
3	2	0	3	4, 5	2, 2
4	4	0	$+\infty$	3	-4
5	2	8	$+\infty$	4	-5

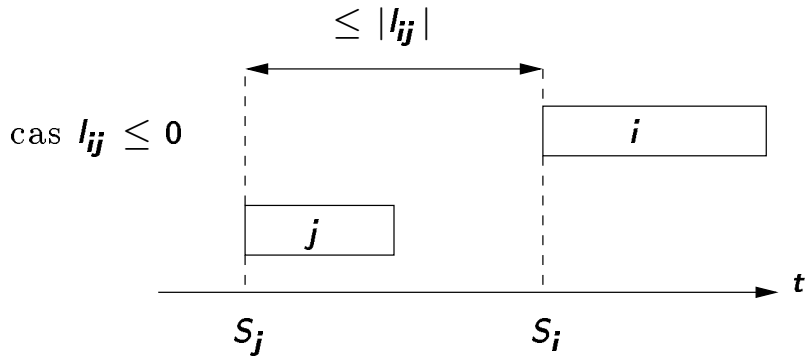
Définition du problème central de l'ordonnancement 2/3

Ordonnancement réalisable

- Application $S : T \rightarrow \mathbb{R}^+$ (S_i date de début de la tâche i) vérifiant :
  - $r_i \leq S_i \leq \tilde{d}_i - p_i$,
  - pour chaque successeur $j \in \sigma(i)$, $S_j \geq S_i + l_{ij}$
- Soit $\mathcal{S}$ l'ensemble des ordonnancements réalisables.

Les deux types de contraintes de succession :



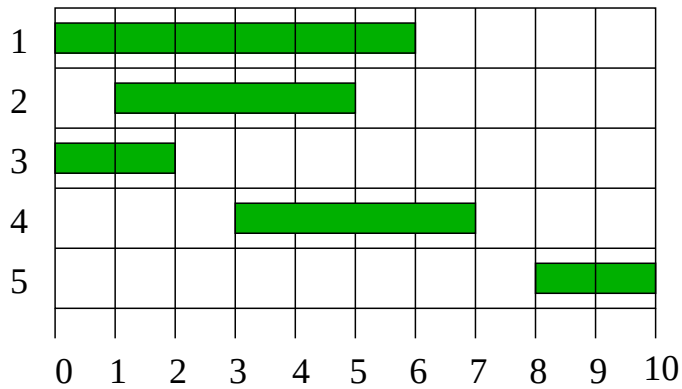


Définition du problème central de l'ordonnancement 3/3

Problème central de l'ordonnancement (PCO)

Trouver un ordonnancement réalisable $S \in \mathcal{S}$ de durée totale $C_{max} = \max_{i \in T}(S_i + p_i)$ minimale.

i	p_i	r_i	$\tilde{d}_i$	$\sigma(i)$	$l_{ij}, j \in \sigma(i)$
1	6	0	$+\infty$	2	1
2	4	0	$+\infty$	1	-3
3	2	0	3	4, 5	2, 2
4	4	0	$+\infty$	3	-4
5	2	8	$+\infty$	4	-5



Problème de potentiels sur un graphe orienté et valué

Définition du graphe $G(V, E, \delta)$

- un ensemble de sommets V,
- un ensemble d'arcs E,
- une fonction de valuation des arcs $\delta : E \longrightarrow \mathbb{R}$,

Potentiel sur le graphe

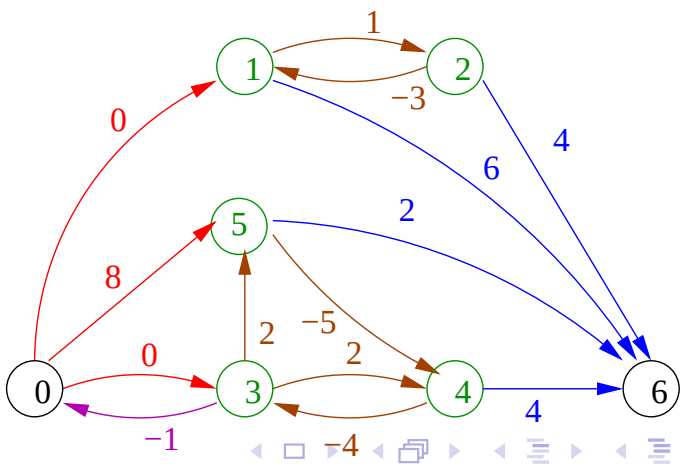
- Une fonction $\pi : T \longrightarrow \mathbb{R}$ est un potentiel sur G si pour tout arc $(i, j) \in E$, $\pi_j \geq \pi_i + \delta_{ij}$
- Soit Π l'ensemble des potentiels.

Equivalence entre potentiels et ordonnancement 1/2

Transformation du PCO en graphe potentiels-tâches

- un ensemble de sommets $V = \{0, 1, \ldots, n, n+1\}$,
- un ensemble d'arcs
  $E = \{(i,j) | i \in T, j \in \sigma(i)\} \cup \{(0,i) | i \in T\} \cup \{(i, n+1) | i \in T\}$,
- une fonction de valuation des arcs $\delta : E \rightarrow \mathbb{R}$:
  - $\delta_{ij} = l_{ij}, \forall i \in T, j \in \sigma(i)$,
  - $\delta_{0i} = r_i, \forall i \in T$
  - $\delta_{i0} = p_i - \tilde{d}_i, \forall i \in T$
  - $\delta_{i,n+1} = p_i, \forall i \in T$

i	p_i	r_i	$\tilde{d}_i$	$\sigma(i)$	$l_{ij}, j \in \sigma(i)$
1	6	6	$+\infty$	2	1
2	4	0	$+\infty$	1	-3
3	2	0	3	4, 5	2, 2
4	4	0	$+\infty$	3	-4
5	2	8	$+\infty$	4	-5



Equivalence entre potentiels et ordonnancement 2/2

Théorème

$\mathcal{S} = \emptyset$ si et seulement si $\Pi = \emptyset$ et il existe une bijection entre l'ensemble des ordonnancement réalisables $\mathcal{S}$ et l'ensemble des potentiels positifs restreints à T.

On a $S_i = \pi_i$, $\forall i \in T$. On pose $S_0 = \pi_0$ et $S_{n+1} = \pi_{n+1}$. On a $C_{max} = S_{n+1}$.

Définitions

Parmi les ordonnancements optimaux, on distingue :

- l'ordonnancement au plus tot ES tel que $ES_i \leq S_i$, $\forall i \in V$, $\forall S \in \mathcal{S}$.
- l'ordonnancement au plus tard LS tel que $LS_i \geq S_i$, $\forall i \in V$, $\forall S \in \{S' \in \mathcal{S} | S'_{n+1} = ES_{n+1}\}$

Résolution du Problème central de l'ordonnancement

Existence d'un ordonnancement réalisable

$\mathcal{S} = \emptyset$ si et seulement si il existe un circuit de longueur positive dans G

Ordonnancement au plus tôt (plus petit potentiel positif)

Si $\mathcal{S} \neq \emptyset$, alors ES_i est égal à la longueur du plus long chemin de 0 à i dans G, pour tout $i \in V$.

Ordonnancement au plus tard (plus grand potentiel tq $S_{n+1} = ES_{n+1}$)

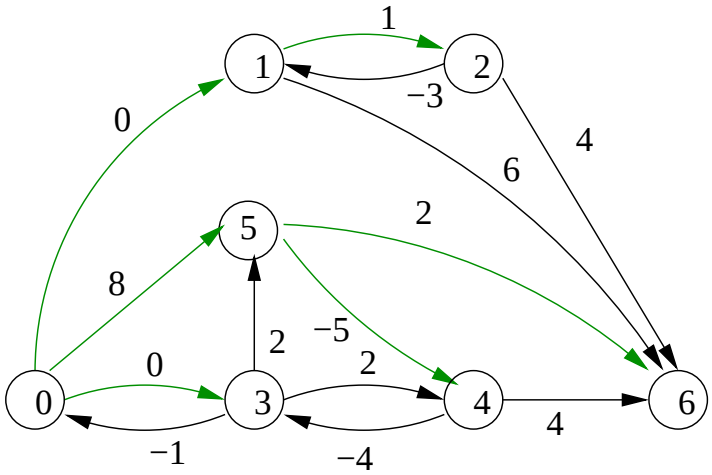
Si $\mathcal{S} \neq \emptyset$, alors LS_i est égal à ES_{n+1} moins la longueur du plus long chemin de i à $n+1$ dans G, pour tout $i \in V$.

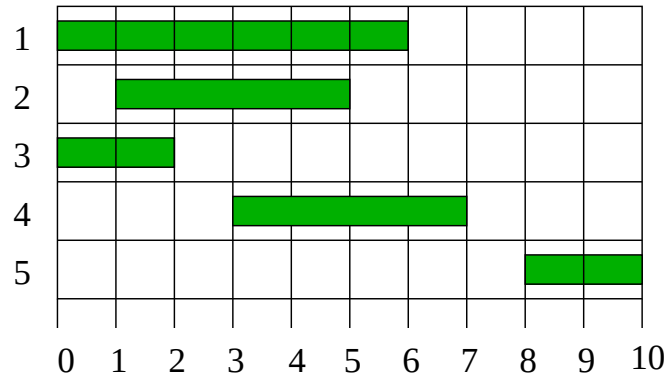
Complexité

L'existence, l'ordonnancement au plus tôt et l'ordonnancement au plus tard peuvent être obtenus en $O(n^3)$ par l'algorithme de floyd-warshall qui calcule les plus long chemins entre toute paire de somments dans G.

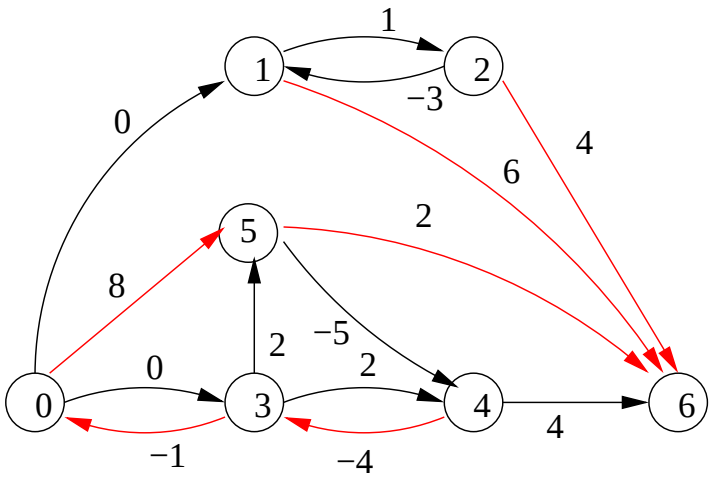
Résolution de l'exemple

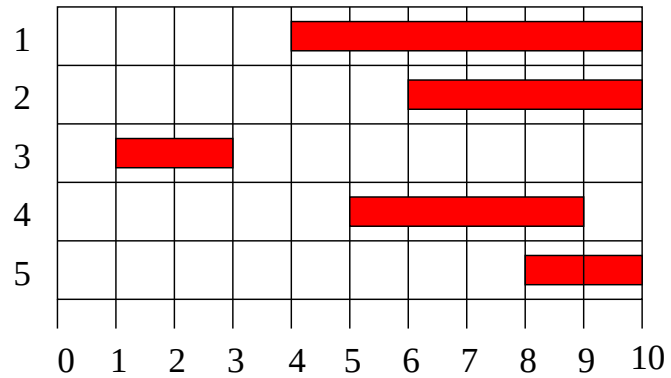
Ordonnancement au plus tôt et plus longs chemins correspondants





Ordonnancement au plus tard et plus longs chemins correspondants





Classification des problèmes d'ordonnancement

La notation $\alpha|\beta|\gamma$

- α décrit le type de ressources et leur organisation
  - $\alpha = 1$: problème à une machine
  - $\alpha = P$: problèmes à machines parallèles
  - $\alpha = F$: problème d'atelier (flow-shop)
  - $\alpha = J$: problème d'atelier (job-shop)
  - $\alpha = PS$: problème à ressources cumulatives (ou discrètes)
- β précise les caractéristiques des tâches
  - $pmtn \in \beta$: possibilité de préemption (interruption et reprise des tâches)
  - $r_i \in \beta$: problème avec dates de disponibilités
  - $\tilde{d}_i \in \beta$: problème avec deadlines
  - $p_i = 1 \in \beta$: problème à durées unitaires
  - $prec \in \beta$: problème à contraintes de précédence simples ($l_{ij} = p_i$)
  - $temp \in \beta$: problème à contraintes de précédence généralisées
- γ décrit la fonction objectif

Types usuels de ressources

Machine unique ou ressource disjonctive $\alpha = 1$, ou J ou F

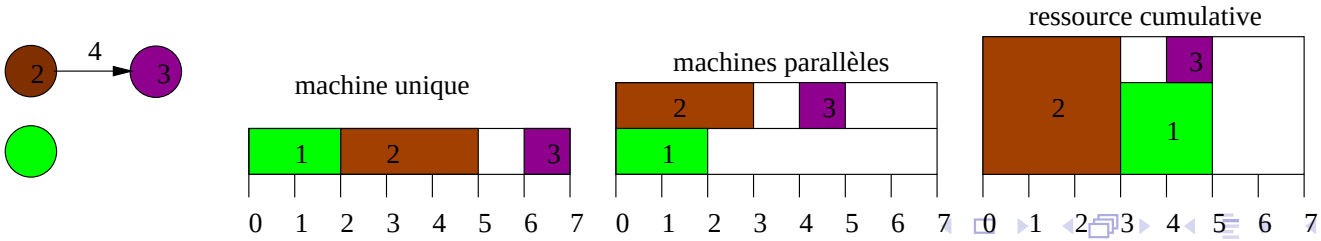
- Chaque tâche i est exécutée par une machine précise m_i
- Une machine ne peut exécuter qu'une tâche à la fois

Machines parallèles $\alpha = P$

- Chaque tâche peut être exécutée par une des machines
- Une machine ne peut exécuter qu'une tâche à la fois

Ressources cumulatives ou discrètes $\alpha = PS$

- Chaque tâche utilise b_{ik} unités de la ressource k
- Une ressource k est disponble à chaque instant en B_k unités



Fonctions objectif usuelles

Maximum ou d'une somme de fonctions élémentaires

La fonction objectif est souvent une composition de fonctions élémentaires f_i des dates de fin des tâches $C_i = S_i + p_i$. On note

- $\gamma = f_{max}$ pour minimiser $\max_{i \in T} f_i(C_i)$
- $\gamma = \sum f_i$ pour minimiser $\sum_{i \in T} f_i(C_i)$
- $\gamma = \sum w_i f_i$ pour minimiser $\sum_{i \in T} w_i f_i(C_i)$ (w_i poids de la tâche i)

Quelques fonctions élémentaires *fi* (d_i date de livraison prévue)

- Date de fin $f_i = C_i$
- Retard algébrique $f_i = L_i = C_i - d_i$
- Avance $f_i = E_i = \max(0, d_i - C_i)$
- Retard vrai $f_i = T_i = \max(0, C_i - d_i)$
- Indicateur de retard $f_i = U_i = \begin{cases} 0 & \text{si } C_i \leq d_i \\ 1 & sinon \end{cases}$

Types d'ordonnancements

Ordonnancements actifs

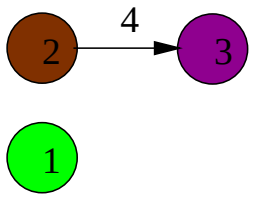
Un ordonnancement S est actif si pour toute tâche i et tout réel $d > 0$ il n'xiste pas d'ordonnancement réalisable S' tel que $S'_i = S_i - d$ et $S'_j = S_j$, $j \neq i$. Soit $\mathcal{S}^a$ l'ensemble des ordonnancements actifs.

Ordonnancements semi-actif (au plus tôt)

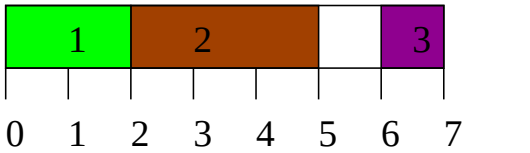
Un ordonnancement S est actif si pour toute tâche i et tout réel $d > 0$ il n'existe pas d'ordonnancement réalisable S' et de réel $\epsilon \leq d$ tel que $S'_i = S_i - \epsilon$ et $S'_j = S_j$, $j \neq i$. Soit $\mathcal{S}^{sa}$ l'ensemble des ordonnancements semi-actifs.

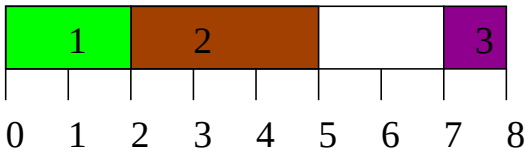
Remarque. Soit $\mathcal{S}^a$ (resp. $\mathcal{S}^{sa}$) l'ensemble des ordonnancements actifs (resp. semi-actifs). On a $\mathcal{S}^a \subseteq \mathcal{S}^{sa} \subseteq \mathcal{S}$.

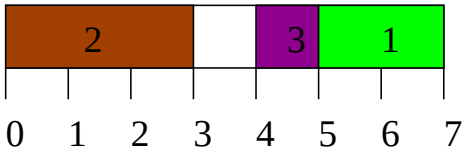
Exemple d'ordonnancements actifs et semi actifs

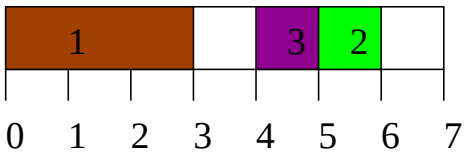


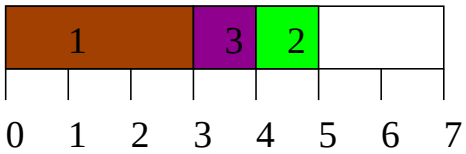
machine unique

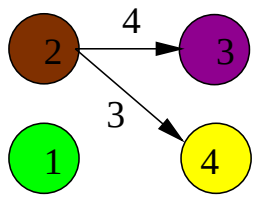




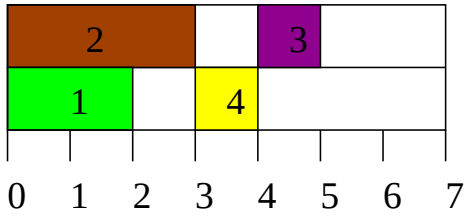


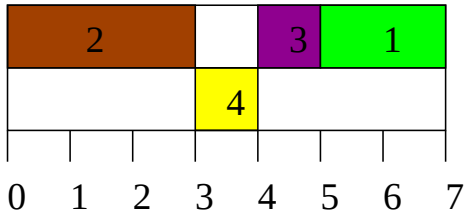


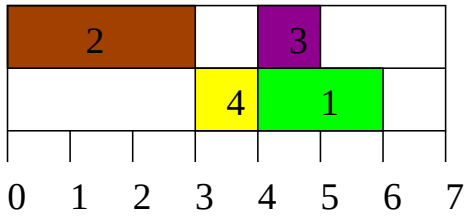


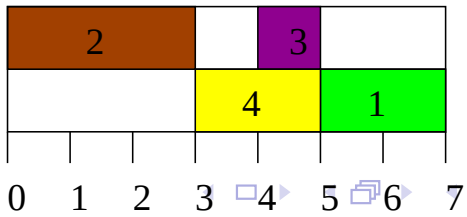


machines parallèles

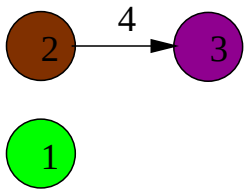






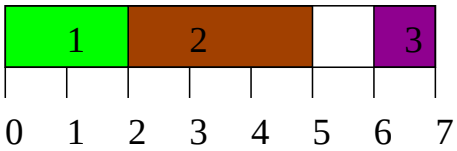


Exemple d'ordonnancements actifs et semi actifs

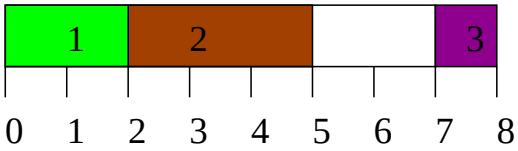


machine unique

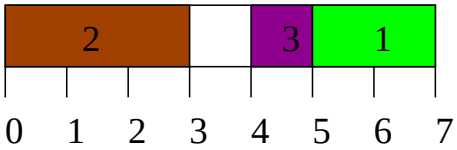
actif



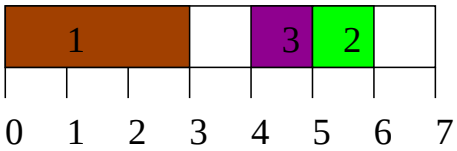
réalisable
non semi−actif



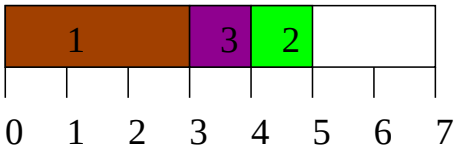
actif

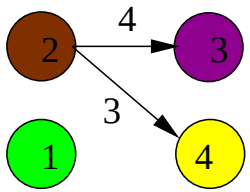


semi−actif
non actif

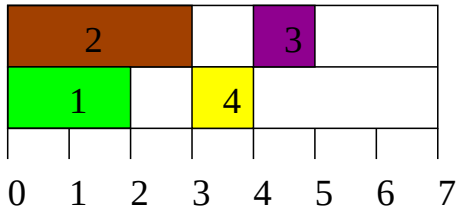


non réalisable!

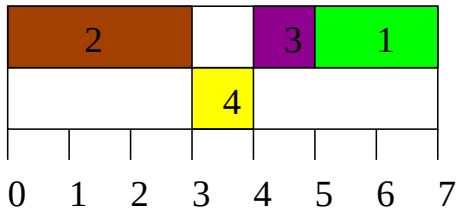




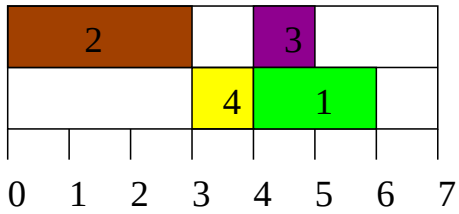
machines parallèles



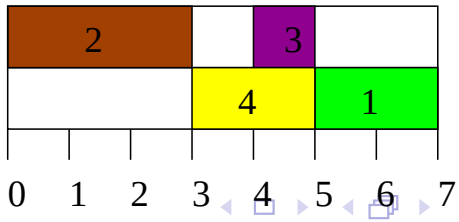
actif



réalisable
non semi−actif



réalisable
non semi−actif
selon la définition



semi−actif
non actif

fonction objectif régulière et ordonnancements dominants

fonction régulière

Une fonction f_i régulière est non décroissante en fonction des dates de fin $C_i = S_i + p_i$. C_i, L_i , T_i U_i sont régulières mais pas E_i.

fonction objectif régulière

$\gamma = f_{\max}, \sum f_i$ ou $\sum w_i f_i$ avec f_i régulière.

Théorème

Les ordonnancements actifs (resp. semi-actifs) sont dominants pour toute fonction objectif régulière : il existe un ordonnancement (resp. semi-actifs) optimal.

Intérêt : diminution de l'espace de recherche.

Problèmes à une machine

Exemples de problèmes de minimisation polynomiaux

- Plus grand retard
- Plus grand retard avec préemption, dates de lancement et précédences
- Fonction régulière de type maximum avec préemption et précédences
- Somme pondérée des dates de fin
- Somme des dates de fin avec préemption et dates de lancement
- Nombre de tâches en retard
- Fonction régulière de type somme avec durées unitaires et dates de lancement

Exemples de problèmes de minimisation NP-difficiles

- Nombre pondéré de tâches en retard
- Plus grand retard avec dates de lancement

Plus grand retard $1|-|L_{max}$

Définition

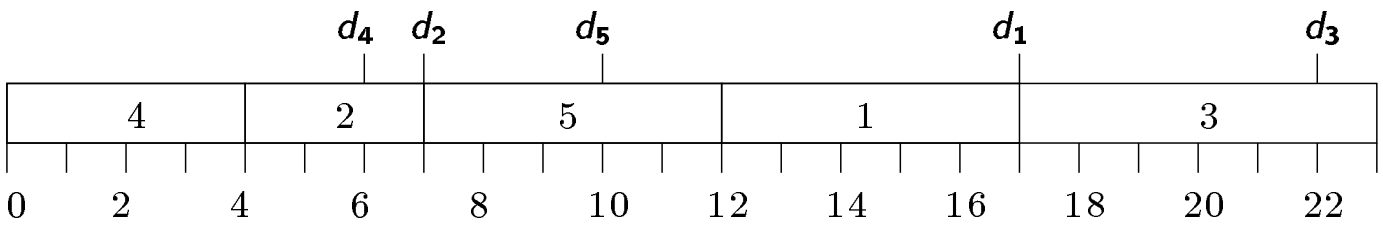
Ordonnancer un ensemble de tâches définies par leurs durées p_i et leurs dates de livraison d_i sur une machine en minimisant le plus grand retard $L_{max} = \max_{i \in T}(C_i - d_i)$.

Résolution (Règle de Jackson, 1955)

Ordonnancer les tâches dans l'ordre des dates de livraison croissante ($O(n \log n)$)

preuve d'optimalité par argument d'échange

i	p_i	d_i
1	5	17
2	3	7
3	6	22
4	4	6
5	5	10



$L_{max} = 2$

Plus grand retard avec préemption, dates de lancement et précédences $1|prec, pmtn, r_i|L_{\max}$ (1/2)

Définition

Ordonnancer un ensemble de tâches définies par des durées p_i, des dates de lancement r_i, des dates de livraison d_i, des contraintes de précédence simples (graphe $G(V, E)$), sur une machine en minimisant le plus grand retard $L_{\max} = \max_{i \in T}(C_i - d_i)$ avec préemption autorisée.

Modification des dates de lancement et de livraison $O(n + |E|)$

Les dates de lancement et de livraison peuvent être mises à jour en utilisant les contraintes de précédence comme suit :

$$r'_i = \max\left(r_i, \max_{i \in \sigma^-(i)}(r'_i + p_i)\right) \quad d'_i = \min\left(d_i, \min_{i \in \sigma(i)}(d'_j - p_j)\right)$$

où $\sigma^-(i)$ ($\sigma(i)$) désignent les prédécesseurs (successeurs) de i.

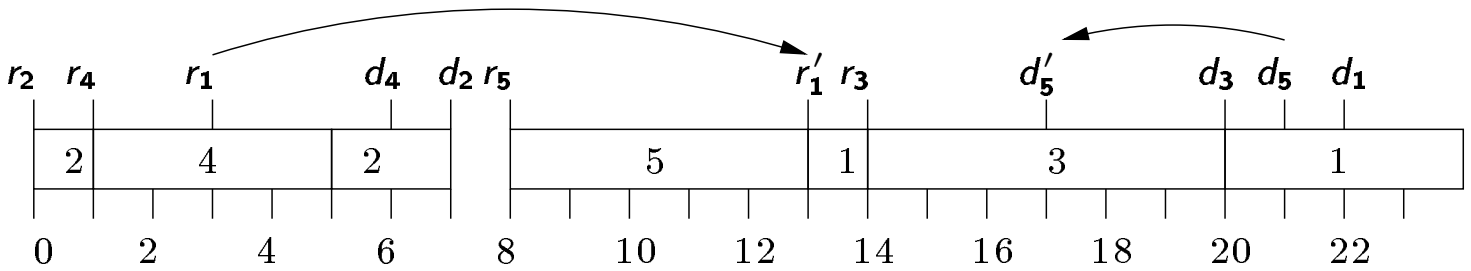
Plus grand retard avec préemption dates de lancement et précédences $1|prec, pmtn, r_i|L_{max}$ $1|prec, pmtn, r_i|L_{max}$ (2/2)

Résolution Algorithme de Jackson préemptif $O(n^2)$

A chaque instant t défini par une date de lancement modifiée ou la fin d'une tâche, ordonnancer la tâche de plus petite date de lancement modifiée en interrompant si nécessaire la tâche en cours à t.

preuve d'optimalité par argument d'échange.

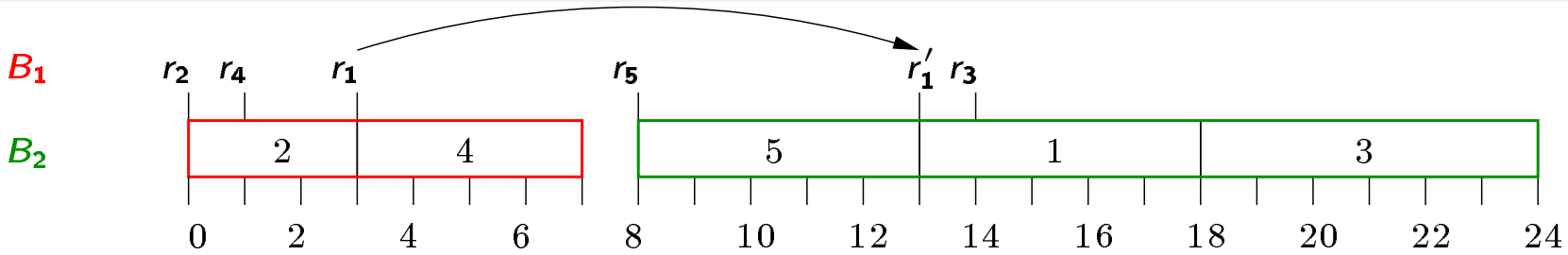
i	p_i	r_i	d_i	$\sigma(i)$
1	5	3	22	–
2	3	0	7	–
3	6	14	20	–
4	4	1	6	–
5	5	8	21	1



Fonction régulière de type maximum avec préemption et précédences $1|prec, pmtn, r_i|f_{\max}$ (1/3)

Blocs de tâches

Soit une solution non prémptive obtenue, en ordonnançant les tâches dans l'ordre croissant des dates de lancement modifiées. La solution obtenue est une succession de blocs de tâches consécutives $B_1, ..., B_b$ séparés par des périodes d'inactivité de la machine.



Théorème (décomposition du problème)

Il existe un ordonnancement préemptif optimal respectant l'ordre des blocs, les tâches étant exécutées sans temps d'inactivité à l'intérieur de chaque bloc. Si $f^*_{\max}(B)$ est la valeur optimale du problème réduit aux tâches de B alors la valeur optimale du problème est $f^*_{\max} = \max_{j=1,...,b} f^*_{max}(B_j)$.

Fonction régulière de type maximum avec préemption et précédences $1|prec, pmtn, r_i|f_{\max}$ (2/3)

Définition récurrente de la solution optimale d'un bloc

Soient :

- $S(B)$ date de début du bloc B.
- $p(B)$ durée totale du bloc B
- $C(B) = S(B) + p(B)$ date de fin du bloc B

Il existe un ordonnancement optimal dans lequel la tâche k sans successeur dans B telle que

$$f_k(C(B)) = \min_{i \in B, \sigma(i) \cap B = \emptyset} f_i(C(B))$$

se termine à $C(B)$. On a donc

$$f^*_{\max}(B) = \max\left(f^*_{\max}(B \setminus \{k\}), f_k(C(B))\right)$$

Fonction régulière de type maximum avec préemption et précédences $1|prec, pmtn, r_i|f_{\max}$ (3/3)

Algorithme récursif

Décompose(T)

1. Déterminer $B_1, \ldots, B_b$
2. $f \leftarrow -\infty$
3. Pour chaque bloc B faire
4. Déterminer k telle que $f_k(C(B)) = \min_{i \in B, \sigma(i) \cap B = \emptyset} f_i(C(B))$
5. $f \leftarrow \max\{f, f_k(C(B), \text{Décompose}(B \setminus \{k\})\}$
6. Retourner f

Résolution par appel de Décompose(T). Complexité $O(n^2)$.

Remarque : cas $r_i = 0$ pour toute tâche i, la solution obtenue est non préemptive : optimal pour le $1|prec|f_{\max}$.

Somme pondérée des dates de fin $1||\sum w_i C_i$

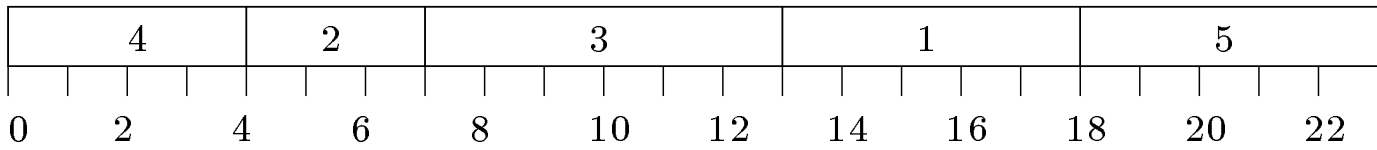
Définition

Ordonnancer un ensemble de tâches définies par des durées p_i, des poids w_i, en minimisant la somme pondérée des dates de fin $\sum w_i C_i = \sum_{i \in T} w_i C_i$.

Résolution (règle du ratio de Smith, 1956)

Ordonnancer les tâches dans l'ordre croissant du rapport p_i/w_i ($O(n \log n)$)

i	p_i	w_i	p_i/w_i
1	5	4	1,25
2	3	3	1
3	6	5	1,2
4	4	5	0,8
5	5	1	5



$$\sum w_i C_i = 4 \times 5 + 7 \times 3 + 13 \times 5 + 18 \times 4 + 23 \times 1 = 201$$

Somme des dates de fin avec préemption et dates de lancement $1|r_i, pmtn|\sum C_i$

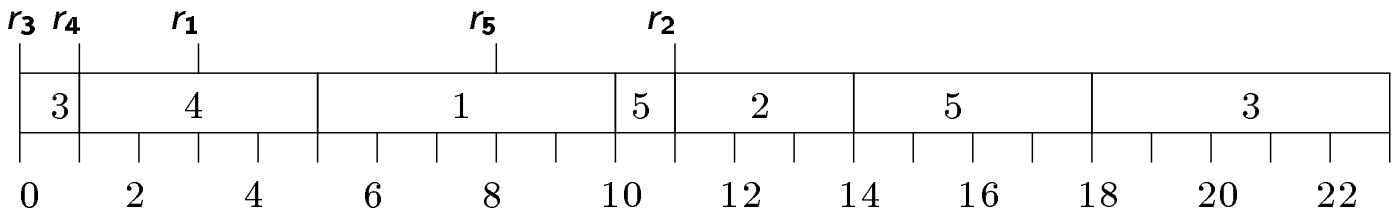
Définition

Ordonnancer sur une machine un ensemble de tâches définies par des durées p_i, des dates de lancement r_i, en minimisant la somme des dates de fin $\sum_{i \in T} C_i$ avec préemption autorisée.

Résolution (règle de Smith modifiée)

At chaque date de début ou fin de tâche, ordonnancer la tâche de plus petite durée restante ($O(n^2)$)

i	p_i	r_i
1	5	3
2	3	11
3	6	0
4	4	1
5	5	8



$\sum C_i = 5 + 10 + 14 + 18 + 3 = 70$

Nombre de tâches en retard $1||\sum U_i$

Définition

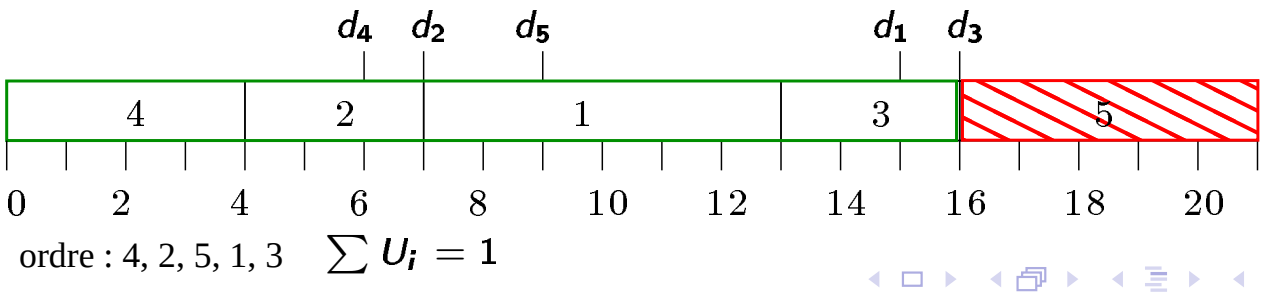
Ordonnancer sur une machine un ensemble de tâches définies par des durées p_i, des dates de livraison d_i, en minimisant le nombre de tâches en retard $\sum_{i \in T} C_i$.

Remarque : avec cette fonction objectif, les tâches en retard peuvent être exécutées arbitrairement après toutes les tâches en avance ou à l'heure.

Définition

Ordonnancer les tâches dans l'ordre croissant des dates de livraison. Dès qu'une tâche est en retard, supprimer des tâches déjà ordonnancées la tâche de plus grande durée.

i	p_i	d_i
1	6	15
2	3	7
3	3	16
4	4	6
5	5	9



Fonction régulière de type somme avec durées unitaires et dates de lancement $1|r_i, p_i = 1|\sum f_i$

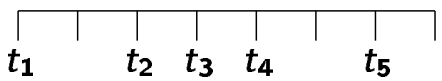
Définition

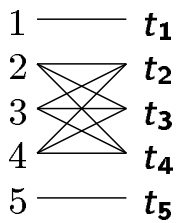
Ordonnancer sur une machine un ensemble de tâches définies par des durées $p_i = 1$, des dates de lancement r_i, en minimisant une fonction régulière de type somme $\sum_{i \in T} f_i$.

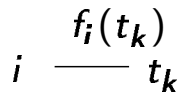
Précalcul des dates d'ordonnancement

Il existe un ordonnancement optimal tel que chaque tâche démarre à une des n dates définies récursivement par $t_1 = r_1$ et $t_i = \max(r_i, t_{i-1} + 1)$ (en supposant $r_1 \leq \ldots \leq r_n$).

i	r_i
1	0
2	3
3	3
4	3
5	6







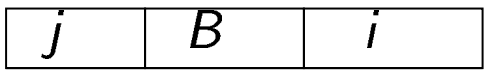
Résolution d'un problème d'affectation ($O(n^3)$)

Somme pondérée du nombre de tâches en retard $1||\sum w_i U_i$ 1/3

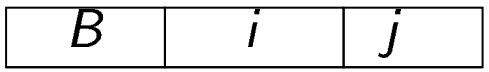
Le problème est NP-difficile (Karp, 1972). Le problème préemptif est donc également NP-difficile.

Caractérisation d'une solution optimale

Supposons $d_1 \leq \ldots \leq d_n$. Il existe un ordonnancement optimal donné par une séquence $i_1, i_2, \ldots, i_s, i_{s+1}, \ldots, i_n$ avec $i_1 < i_2 < \ldots < i_s$ où toutes les tâches $i_1, \ldots, i_s$ sont à l'heure et toutes les tâches $i_s + 1, \ldots, i_n$ sont en retard.



$d_i \leq d_j$



échange réalisable

Somme pondérée du nombre de tâches en retard $1||\sum w_i U_i$ 2/3

Résolution par programmation dynamique.

Plongement dans une famille de sous-problèmes :
On définit $F_j(t)$ pour $j = 1, \ldots, n$ et $t = 0, \ldots, P = \sum_{i=1}^{n} p_i$, la valeur minimale du critère pour le problème d'ordonnancement des tâches $1, \ldots, j$ sachant que l'ordonnancement des tâches à l'heure se termine avant la date t. La valeur de la solution optimale est $F_n(\min(d_n, P))$.

Définition récurrente de $F_j(t)$

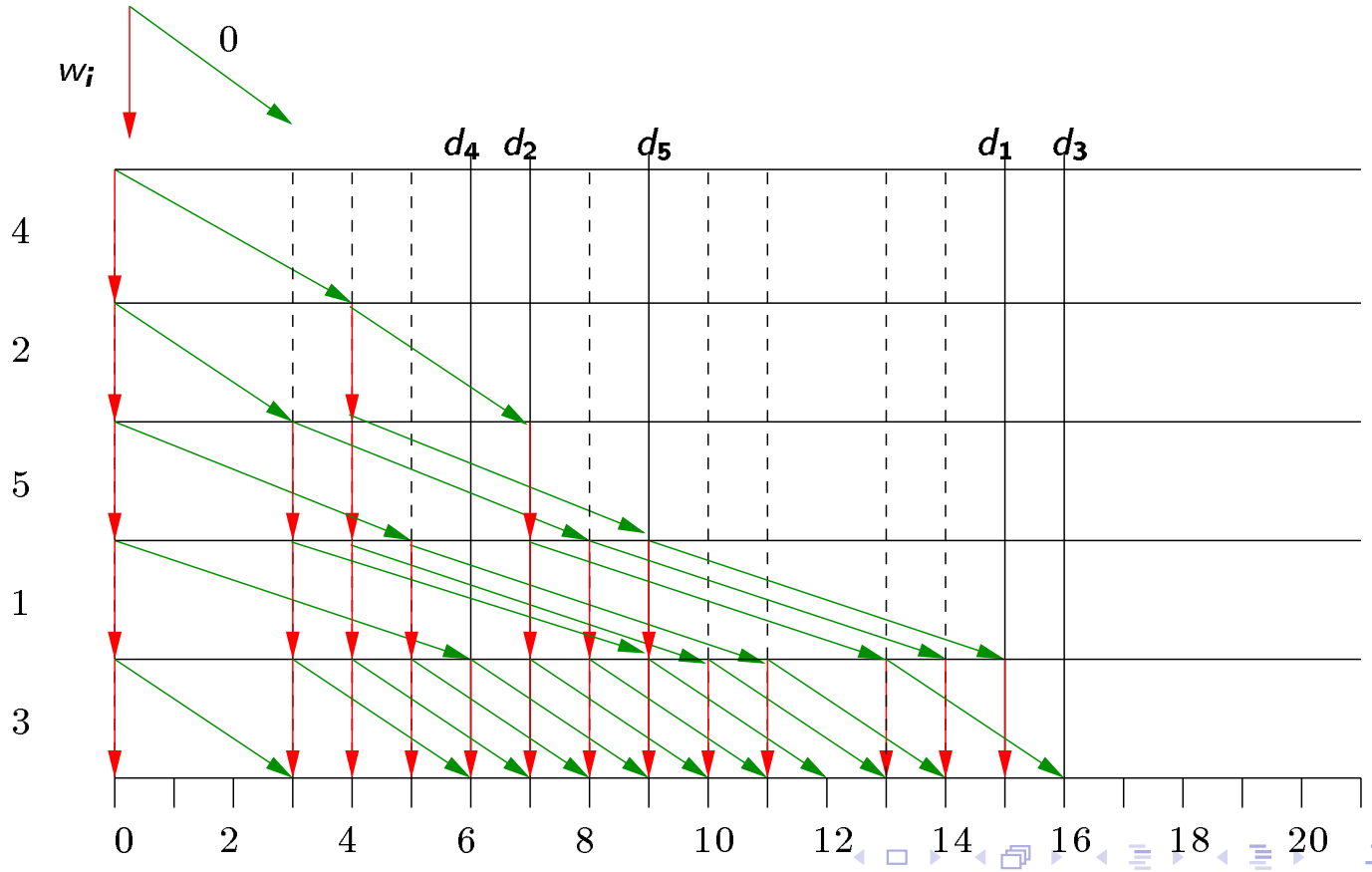
- Si $0 \leq t \leq d_i$ et $j \geq 1$ alors soit j est à l'heure à t et donc elle peut être placée à la fin de la séquence, soit j est en retard donc $F_j(t) = \min(F_{j-1}(t - p_j), F_{j-1}(t) + w_j)$.
- Si $t > d_j$ et $j \geq 1$ alors $F_j(t) = F_j(d_j)$ car toutes les tâches de $1, \ldots, j$ se terminant après d_j sont en retard.
- Si $t < 0$ ou $j = 0$ alors $F_j(t) = 0$

Somme pondérée du nombre de tâches en retard $1||\sum w_i U_i$ 2/3

Calcul ascendant des $F_j(t)$: $O(n \sum_{i=1}^{n} p_i)$ (algorithme pseudo polynomial).

Visualisation graphique de l'algorithme de programmation dynamique comme un plus court chemin

i	p_i	d_i
1	6	15
2	3	7
3	3	16
4	4	6
5	5	9



Plus grand retard avec dates de lancement $1|r_i|L_{max}$ 1/3

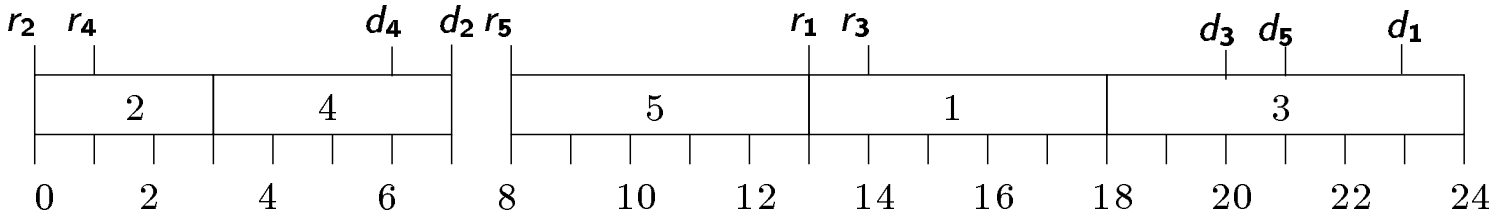
Le problème est NP-difficile (Lenstra et al. 1977).

Résolu efficacement par la procédure de séparation et d'évaluation de Carlier, 1982, basée sur l'analyse d'une solution approchée.

Algorithme de Jackson Non Préemptif (JNP)

1. $t \leftarrow \min_{i \in T} r_i$
2. Sélectionner la tâche i telle que $r_i \leq t$ de plus petite date de livraison et l'ordonnancer à t.
3. décaler t à la première date où une tâche non-ordonnancée est disponible et aller à l'étape 2.

i	p_i	d_i
1	5	23
2	3	7
3	6	20
4	4	6
5	5	21

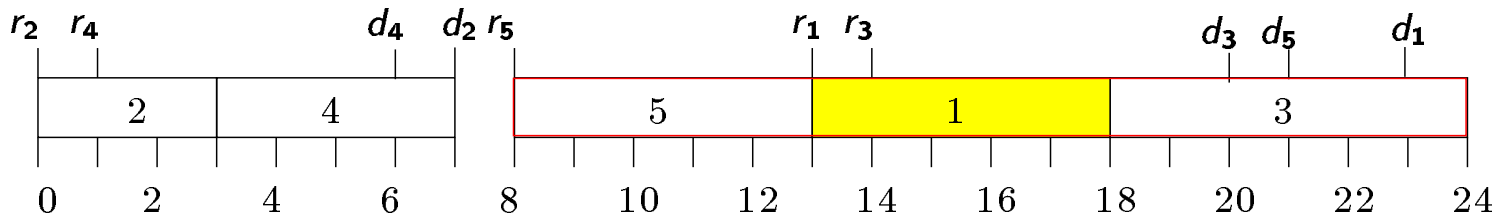


solution par l'algorithme de Jackson non préemptf $L_{max} = 4$

Plus grand retard avec dates de lancement $1|r_i|L_{\max}$ 2/3

Analyse de la solution de Jackson non préemptive

Soit $i_1, \ldots, i_p$ le "chemin critique" tel que $L_{\max} = r_1 + \sum_{k=1,\ldots,p} p_{i_k} - d_{i_p}$. Si pour tout $k = 1, \ldots, p-1$, $d_{i_k} \leq d_{i_p}$, la solution est optimale. Sinon, soit c le plus grand indice tel que $d_{i_c} > d_{i_p}$. Soit de plus $J = \{i_c + 1, \ldots, i_p\}$ Dans toute solution optimale i_c est exécuté, soit avant, soit après toutes les tâches de J.



Bloc critique $\{5, 1, 3\}$ tâche critique 1 $J = \{3\}$

Plus grand retard avec dates de lancement $1|r_i|L_{max}$ 3/3

Algorithme de Carlier, 1982

1. $BS \leftarrow +\infty$
2. $Q \leftarrow$Empiler(r, p, d)
3. Tant que $Q \neq \emptyset$ Faire
4. $\quad (r, p, d) =$Dépiler(Q)
5. $\quad$ Calculer i_c et J par JNP et Mettre à jour BS si nécessaire.
6. $\quad$ Si la condition d'optimalité est vérifiée retourner JNP.
7. $\quad$ Calculer BI par l'algorithme de Jackson préemptif.
8. $\quad$ Si $BI \leq BS$ alors
9. $\quad\quad r' \leftarrow r$; $r'_{i_c} \leftarrow \max(r_{i_c}, \min_{i \in J} r_i + \sum_{i \in J} p_i)$
10. $\quad\quad d' \leftarrow d$; $d'_{i_c} \leftarrow \min(d_{i_c}, \max_{i \in J} d_i - \sum_{j \in J} p_i)$.
11. $\quad\quad Q \leftarrow$Empiler(r, p, d')
12. $\quad\quad Q \leftarrow$Empiler(r', p, d)

Problèmes à machines parallèles

Exemples de problèmes de minimisation polynomiaux

- Problème de minimisation de la durée totale avec préemption
- Problème de solution réalisable avec dates de lancement, dates d'échéance et préemption

Problème de minimisation de la durée totale avec préemption $P|pmtn|C_{max}$

Définition

Ordonnancer sur m machines parallèles, n tâches de durées p_i en minimisant la durée totale $C_{max} = \max_{i \in T} C_i$, avec préemption autorisée.

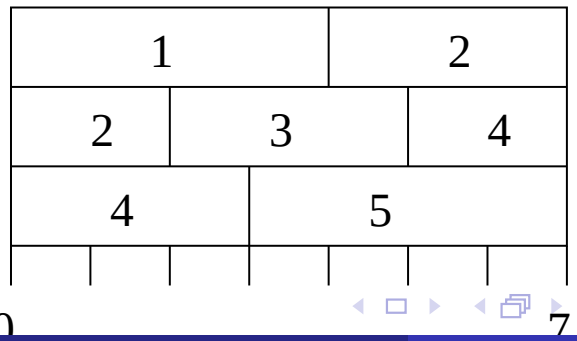
La valeur exacte de la solution optimale peut être calculée directement :

$$C^*_{max} = \max \left\{ \max_{i \in T} p_i, (\sum_{i=1}^{n} p_i)/m \right\}$$

La solution se construit en $O(n)$ en remplissant les machines en prenant les tâches dans n'importe quel ordre à partir de la machine 1. Dès que C^*_{max} est atteint, la tâche est interrompue et reprise sur la machine suivante.

$m = 3$

i	p_i
1	4
2	5
3	3
4	5
5	4

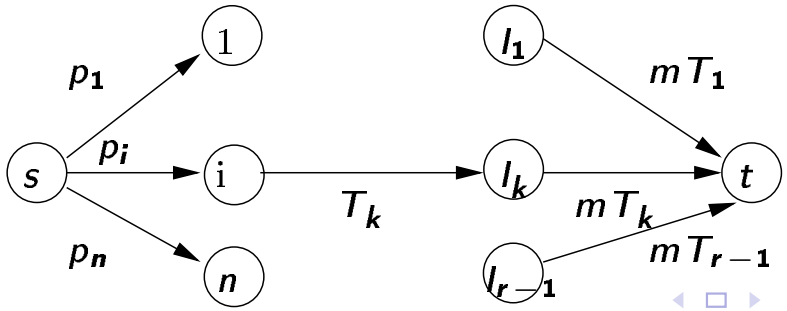


Problème de solution réalisable avec dates de lancement, dates d'échéance et préemption $P|pmtn, r_i, \tilde{d}_i|-$

Définition

Ordonnancer sur m machines parallèles, n tâches de durées p_i, de dates de lancement r_i et dates d'échéances $\tilde{d}_i$, avec préemption autorisée.

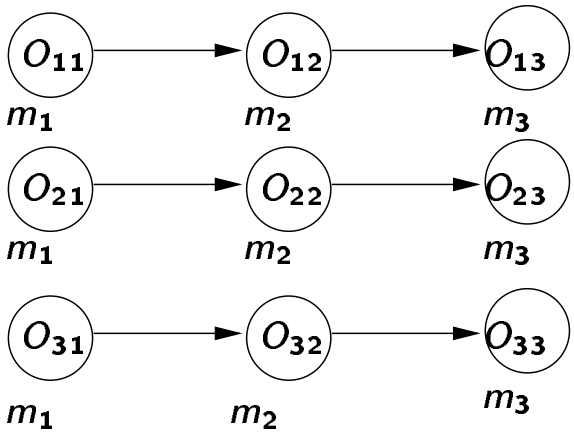
Le problème peut être transformé en problème de flot maximal. Soit $t_1 < t_2 < \ldots < t_r$ la séquence des valeurs différentes des r_i et des d_i. On considère les intervalles $I_k = [t_k, t_{k+1}]$ de longueur $T_k = t_{k+1} - t_k$. On crée un sommet par tâche et par intervalle ainsi qu'un sommet source s et un sommet puit t. On définit un arc de capacité T_k entre une tâche i et un intervalle tel que $r_i \leq t_k$ et $t_{k+1} \leq d_i$. On définit un arc de capacité p_i entre s et i et un arc de capacité mT_k entre I_k et t. Il existe une solution au problème $P|pmtn, r_i, \tilde{d}_i|-$ si et seulement s'il existe un flot maximal de valeur $\sum_{i=1}^{n} p_i$.



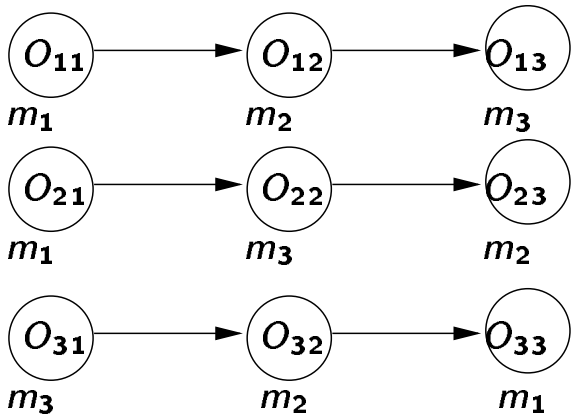
Problèmes d'atelier

Définition

Dans les problèmes d'atelier les tâches sont regroupées en travaux J_i, $i = 1, \ldots, n$. Un travail est une succession d'opérations (les tâches) liées par des contraintes de précédence sous la forme d'une chaine. Chaque opération O_{ij} pour l'opération j du travail i est définie par sa durée p_{ij} et la machine sur laquelle elle est exécutée m_i. On distingue différents problèmes d'atelier selon l'ordre de visite des machines par les opérations. L'objectif est souvent de minimiser la durée totale de la réalisation des travaux.



Flow-Shop

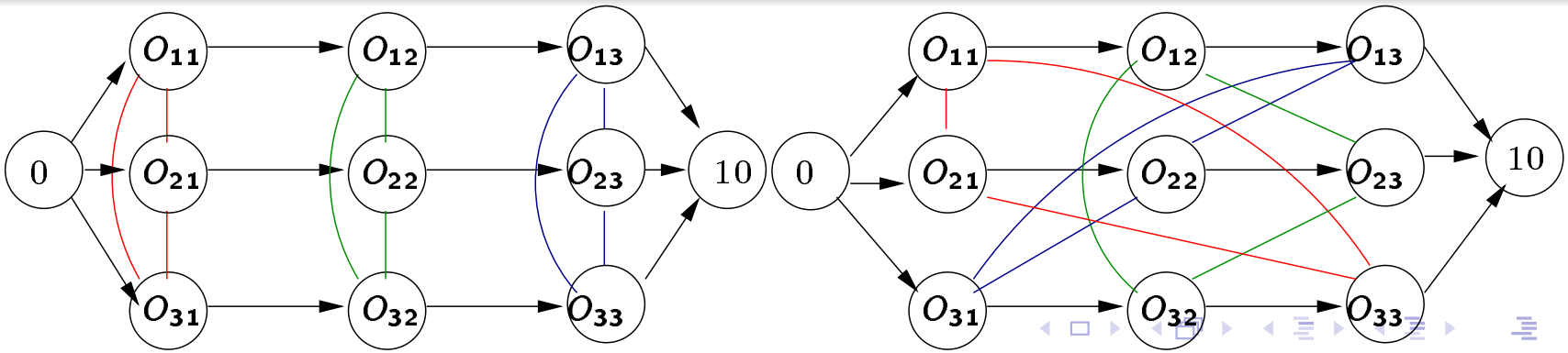


Job-Shop

.

Représentation par un graphe disjonctif

Le graphe représente les contraintes de précédence et les possiblités de séquencement sur les machines. On a un sommet par opération et un sommet de début et de fin de l'ordonnancement. En plus des contraintes de précédences définies par les travaux, on a des arêtes représentant le partage des machines. L'orientation d'une arête donne un arc (O_{ij}, O_{kl}) qui représente la décision d'ordonnancer O_{ij} avant O_{kl} sur la machine. Sachant que les arcs sont valués par la durée de l'opération origine, le problème d'atelier peut s'énoncer comme la recherche d'une orientation totale des arêtes de manière à minimiser la longueur du plus long chemin entre le sommet de début et le sommet de fin.



Problèmes d'atelier

Les problèmes d'atelier (ou disjonctifs) sont en général NP-difficile.

Exemple de problème

- Flow-shop à deux machines (polynomial)
- Job-shop (NP-difficile)

Problèmes de Flow-shop à deux machines $F2||C_{max}$

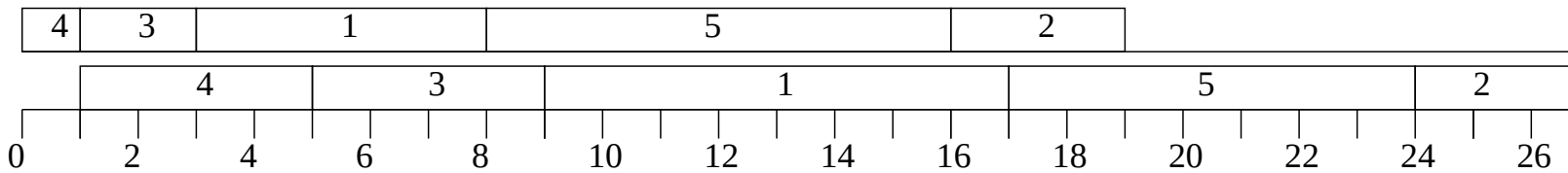
Ordonnancements de permutation

Dans le flow-shop à deux machines, les ordonnancements de permutation, pour lesquels la séquence des opérations est la même sur les deux machines, sont dominants. Une solution peut donc être représentée par une liste de travaux.

Résolution par l'algorithme de Johnson, 1954

On construit la liste des travaux en construisant progressivement une liste de gauche G puis une liste de droite D. A chaque étape, on cherche l'opération O_{ij} de plus petite durée. Si $j = 1$ on ajoute i à la fin de G. Sinon on l'insère au début de D. On concatène ensuite G et D.

i	p_{i1}	p_{i2}
1	4	8
2	3	3
3	3	4
4	1	4
5	8	7



Problème de Job-shop $J||C_{max}$

Le problème est NP-difficile. On trouve pour le résoudre de nombreuses méthodes approchées (heuristiques et métaheuristiques) et des méthodes exactes de type procédure de séparation et évaluation.

Principe des méthodes de séparation et d'évaluation pour le Job-Shop

- Séparation. Les schémas de branchement les plus efficaces sont basées sur l'analyse du chemin critique dans le graphe disjonctif orienté. Le chemin critique est décomposé en blocs de taches consécutives sur la même machine. Pour améliorer une solution, il est nécessaire qu'au moins une tâche d'un bloc soit ordonnancée avant la première tâche du bloc ou après la dernière, ce qui sépare l'espace des solutions.
- Evaluation. Les méthodes d'évaluation les plus efficaces sont basées sur les méthodes de **propagation de contraintes**. Ces méthodes, connues sous le nom de sélections immédiates, edge-finding, etc. permettent de réduire les fenêtres de temps des tâches en déduisant des contraintes de précédence entre tâches utilisant la même machine.

Problème d'ordonnancement de projet avec contraintes de ressources $PS|prec|C_{max}$

Le problème est NP-difficile. On trouve pour le résoudre de nombreuses méthodes approchées (heuristiques et métaheuristiques) et des méthodes exactes de type procédure de séparation et évaluation.