Ordonnancement

Christian Artigues

LAAS-CNRS

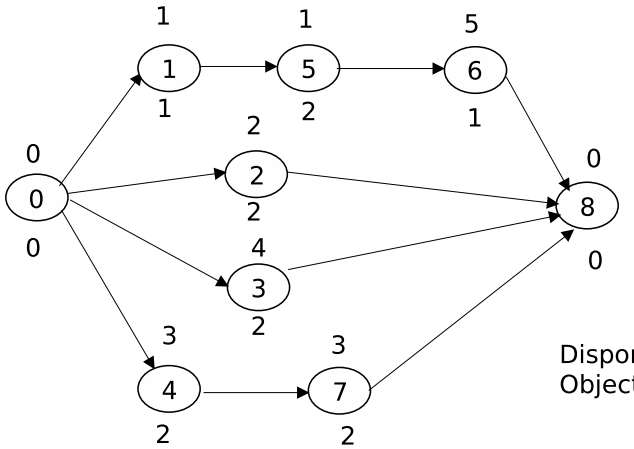
ISAE-Supaéro - FISA

01/02/2022

L'ordonnancement de tâches

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.



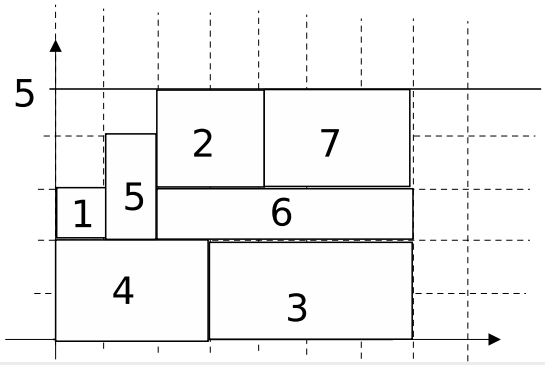
duree(i)

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

- Gestion de la production
- Gestion de projet
- Planification de personnel
- Allocation de ressources dans les télécommunications
- Systèmes d'exploitation des ordinateurs

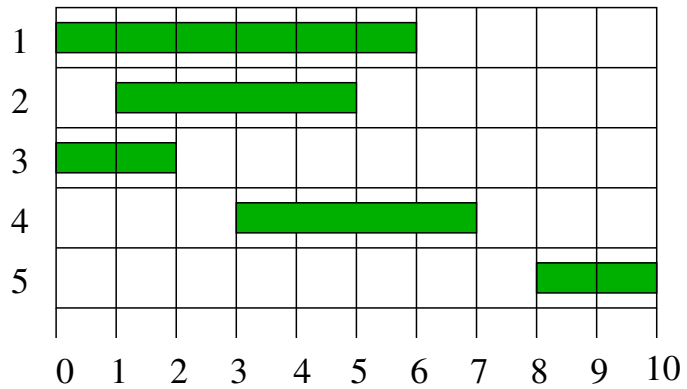
Le problème central de l'ordonnancement

Problème : Déterminer les dates de débuts d'un ensemble de n tâches $\mathcal{J}$ soumises à des contraintes de précédences Γ où $i \in \mathcal{J}$ est définie par

- Durée p_i
- Date de lancement r_i
- Deadline $\tilde{d}_i$
- Ensemble de successeurs Γ_i avec un délai $\delta_{i,j}$, $\forall j \in \Gamma_i$

Objectif : minimiser la durée du projet (plus grande date de fin des tâches)

i	p_i	r_i	$\tilde{d}_i$	Γ_i	$\delta_{i,j}, j \in \Gamma_i$
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

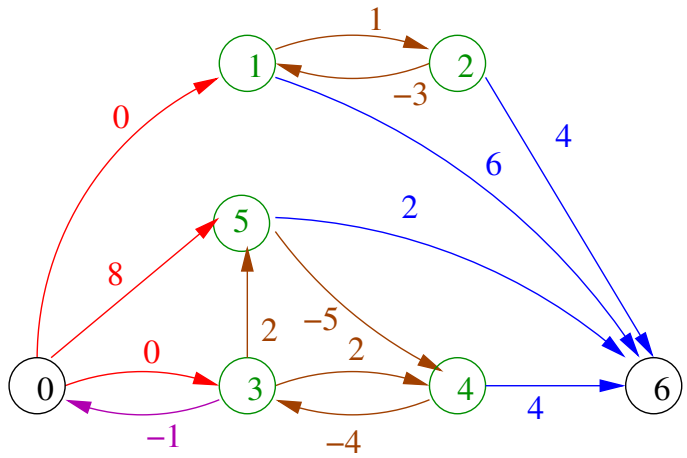


Le graphe conjonctif ou potentiels-tâches 1/2

On construit le graphe $G = (V, A, l)$ comme suit :

- Ensemble des sommets : $V = \mathcal{J} \cup \{0, n+1\}$
- Ensemble des arcs :
  $A = \{l(0,i), (i,0), (i,n+1) | \forall i \in \mathcal{J}\} \cup \{(i,j) | \forall i \in \mathcal{J}, \forall j \in \Gamma_i\}$
- Valuations des arcs : $\forall i \in \mathcal{J}$: $l_{0,i} = r_i$, $l_{i,0} = p_i - \tilde{d}_i$, $l_{i,n+1} = p_i$, et $l_{i,j} = \delta_{i,j}$, $\forall j \in \Gamma_i$

i	p_i	r_i	$\tilde{d}_i$	$\Gamma(i)$	$l_{ij}, j \in \sigma(i)$
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



Le graphe conjonctif ou potentiels-tâches 2/2

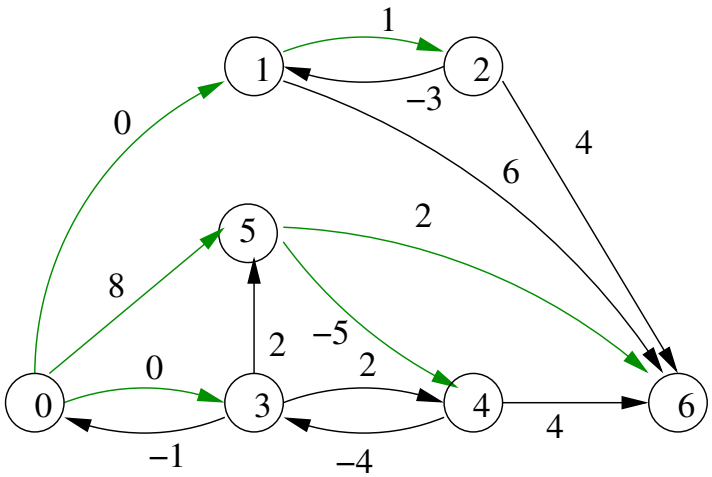
Propriétés remarquables

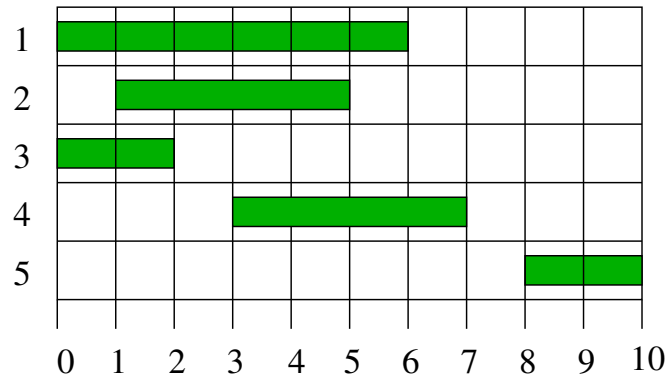
Soit $\mathcal{S}$ l'ensemble des ordonnancements réalisables

- Le problème est irréalisable $\mathcal{S} = \emptyset$ si et seulement s'il existe un circuit de longueur positive dans G
- La date de début au plus tôt $ES_i = \min_{S \in \mathcal{S}} S_i$ est le plus long chemin de 0 à i
- La date de début au plus tard $LS_i = \max_{S \in \mathcal{S} | S_{n+1} = ES_{n+1}} S_i$ est ES_{n+1} moins la longueur du plus long chemin de i à $n+1$

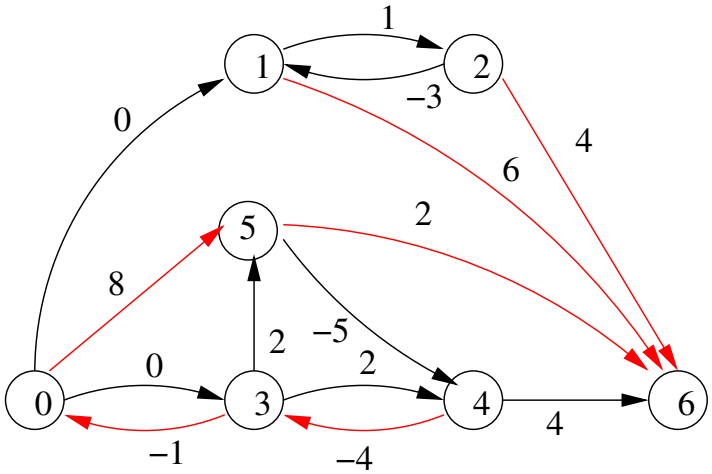
Ordonnancements au plus tôt/tard, chemin critique

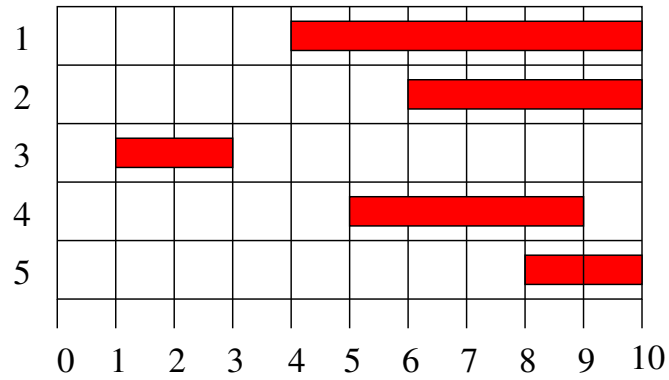
Ordonnancement au plus tôt et plus longs chemins correspondants





Ordonnancement au plus tard et plus longs chemins correspondants





Résolution

Formulation primale

$\min\ S_{n+1}$ s.c.
$S \in \mathcal{S} =$
$\{S_j - S_i \geq l_{i,j}, (i,j) \in E\}$

Problème de *potentiels* sur les sommets dans un graphe potentiel-tâches.

Formulation duale

$\max \sum_{(i,j)\in E} l_{i,j}\phi_{i,j}$ s.c.
$\Phi \in \mathcal{D} =$

$$\left\{ \begin{array}{l} \sum\limits_{(i,j)\in E} \phi_{i,j} = \sum\limits_{(i,j)\in E} \phi_{j,i}, i \in \{1,...,n\} \\ \sum\limits_{(i,n+1)\in E} \phi_{i,n+1} = 1 \\ \phi_{i,j} \geq 0, (i,j) \in E \cup (n+1,0) \end{array} \right\}$$

Problème de *plus long chemin* entre 0 et $n+1$.

- Formulation récurrente $S_0^0 = 0$, $S_j^0 = -\infty$, $j \in V \setminus \{0\}$ et $S_j^k = \max(S_j^{k-1}, \max_{i\in V,(i,j)\in A} S_i^{k-1} + l_{i,j})$, $j \in V$, $k \geq 1$

  Se résout par programmation linéaire ou bien par l'algorithme de Bellman-Ford en $O(|V||A|)$.

Algorithme de Bellman-Ford

Calcule les $(ES_j)_{j \in V}$: date de début au plus tôt de j (la longueur du plus long chemin entre 0 et j) et mémorise le plus long chemin dans $(\gamma_j)_{j \in V}$: prédécesseur de j sur le plus long chemin entre 0 et j.

$S_0 \leftarrow 0$, $S_j \leftarrow -M$, $j \in V \setminus \{0\}$
Pour $k = 1$ à $|V| - 1$ **faire**
  Pour chaque arc $(i, j) \in A$ **faire**
    Si $S_j < S_i + l_{i,j}$ **alors**
    $S_j \leftarrow S_i + l_{i,j}; \gamma_j \leftarrow i$
    Fin Si
  Fin Pour
Fin Pour

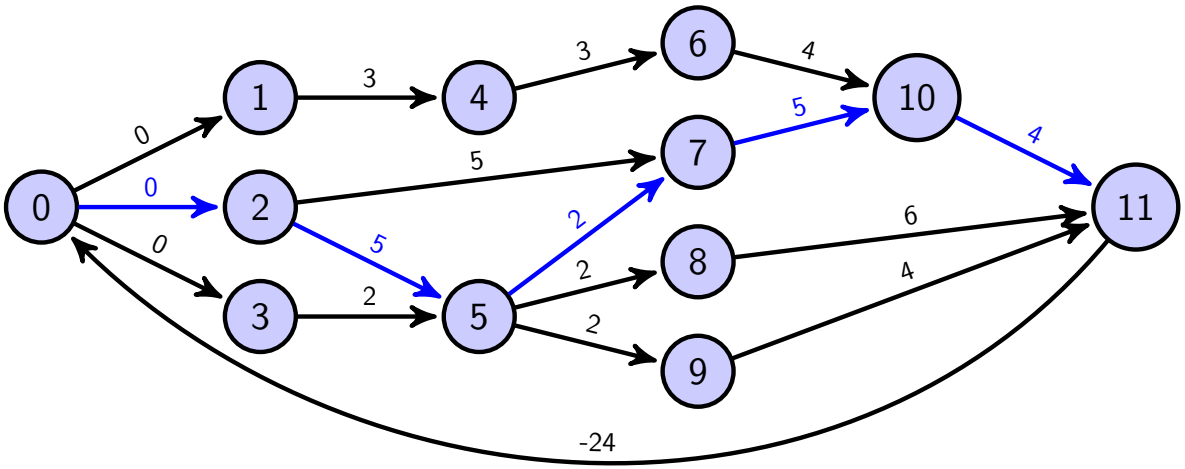
EXERCICE

Exercice 1

- Ecrire les programmes linéaires dual et primal pour l'exemple à 5 tâches
- Calculez les dates au plus tôt et au plus tard des tâches par énumération des chemins
- Calculez les dates au plus tôt et au plus tard des tâches par l'algorithme de Bellman-Ford.
- Calculez les chemins critiques

Cas particulier, le problème d'ordonnancement de projet

Lorsque on a $\delta_{i,j} = p_i$, $\forall (i,j) \in \Gamma$ et pas de deadlines, on parle d'ordonnancement de projet standard (CPM, PERT, MPM).
Dans ce cas, il ne peut y avoir de cycles dans le graphe et un tri topologique des sommets permet de faire la récursion en $O(|A|)$.



$$T = 24$$

i	p_i	$[ES, LS]$
1	3	$[0, 10]$
2	5	$[0, 8]$
3	1	$[0, 12]$
4	3	$[3, 13]$
5	2	$[5, 13]$
6	4	$[6, 16]$
7	5	$[7, 15]$
8	6	$[7, 18]$
9	4	$[7, 20]$
10	4	$[12, 20]$
11	0	$[16, 24]$

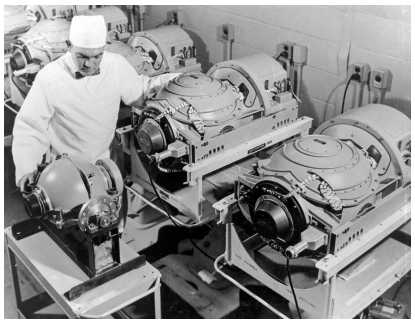
EXERCICE

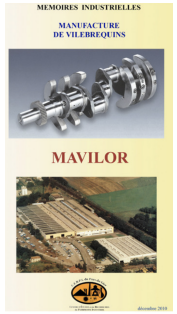
Exercice 2

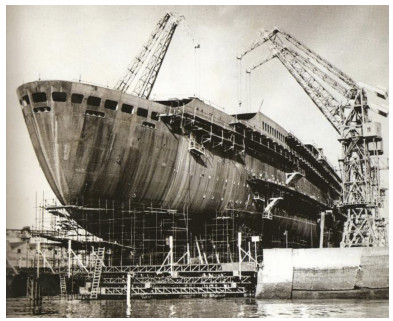
- Calculez les dates au plus tôt et au plus tard des tâches par énumération des chemins
- Calculez les dates au plus tôt et au plus tard des tâches par l'algorithme de Bellman-Ford simplifié.

Un peu d'histoire et d'applications prestigieuses

- Méthode CPM (Critical Path Method). Inventée par James E. Kelley (Remington Rand) et Morgan R. Walker entre 1956 et 1959 (Dupond). Utilisée en 1966 pour la construction des tours jumelles du World Trade Center.
- Méthode PERT (Program Evaluation & Review Technique) Inventée par Booz Allen Hamilton et la U.S. Navy pour le développement des missiles Polaris en 1958
- Méthode MPM (Méthodes des Potentiels Métra). Inventée par Bernard Roy en 1958 pour la société Sema et la planification de la production de villebrequins de l'Usine Mavilor puis de l'équipement du paquebot France.







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

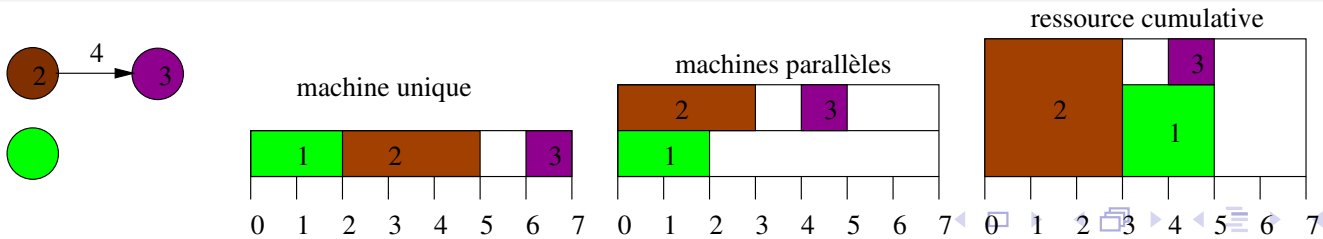
- Chaque tâche i est exécutée par une machine précise m_i
- Une machine ne peut exécuter qu'une tâche à la fois

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

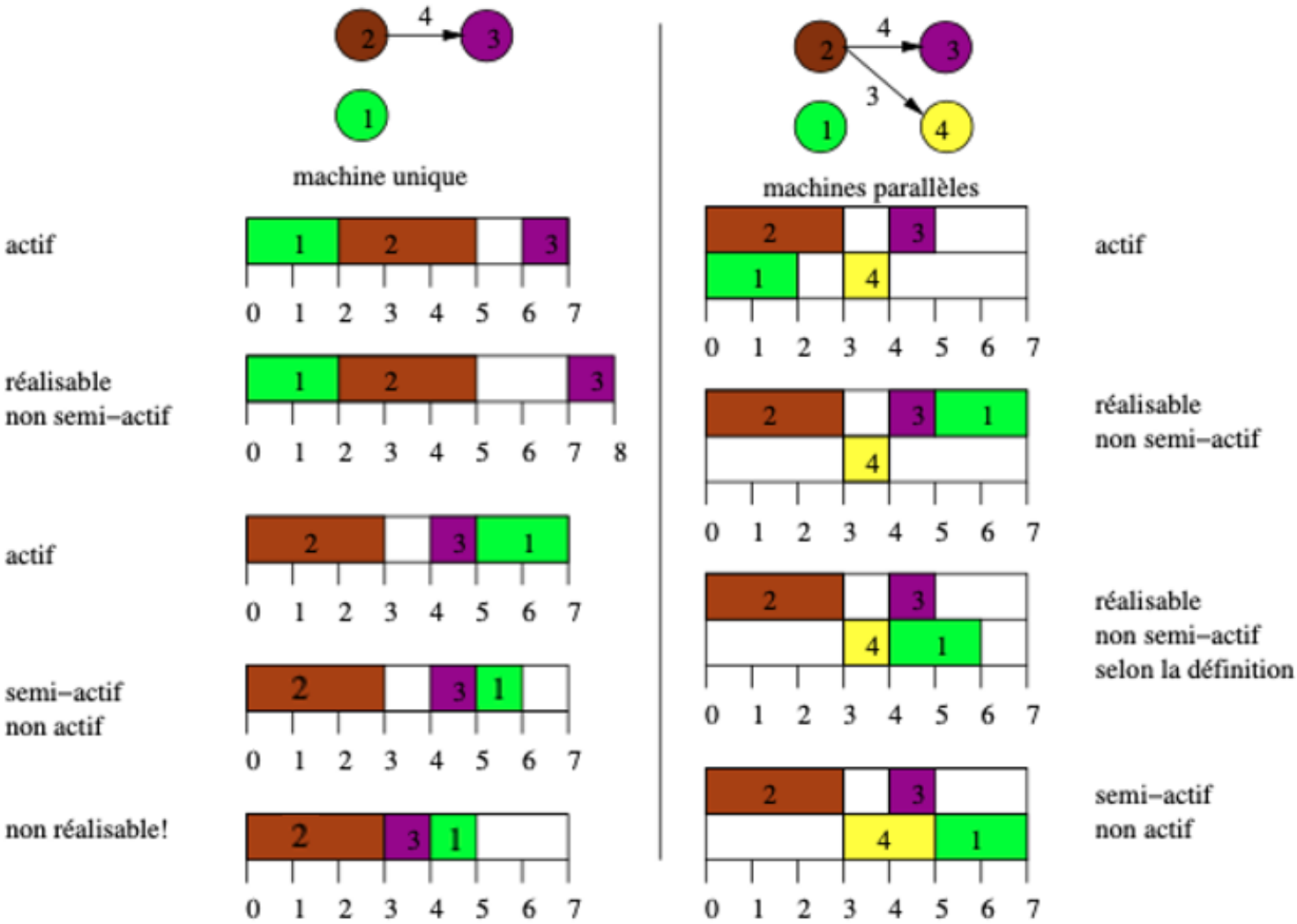
Un ordonnancement S est actif si pour toute tâche i et tout réel $d > 0$ il n'xiste pas d'ordonnancement réalisable S' tel que $S'_i = S_i - d$ et $S'_j = S_j$, $j \neq i$. Soit $\mathcal{S}^a$ l'ensemble des ordonnancements actifs.

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 $\mathcal{S}^a \subseteq \mathcal{S}^{sa} \subseteq \mathcal{S}$.

Exercice 3 : ordonnancements réalisables, actifs, semi actifs ?



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

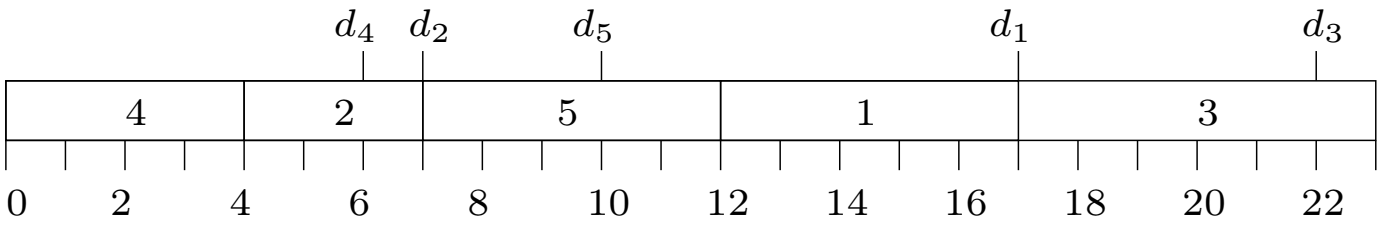
Ordonnancer un ensemble de tâches définies par leurs durées p_i et leurs dates de livraison d_i sur une machine en minimisant le plus grand retard $L_{\max} = \max_{i \in T}(C_i - d_i)$.

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



$L_{\max} = 2$

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

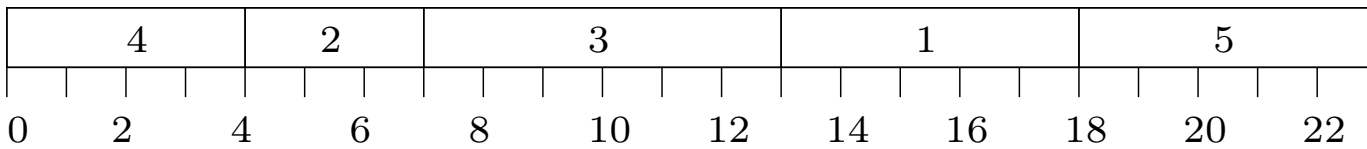
Définition

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



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

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

Définition

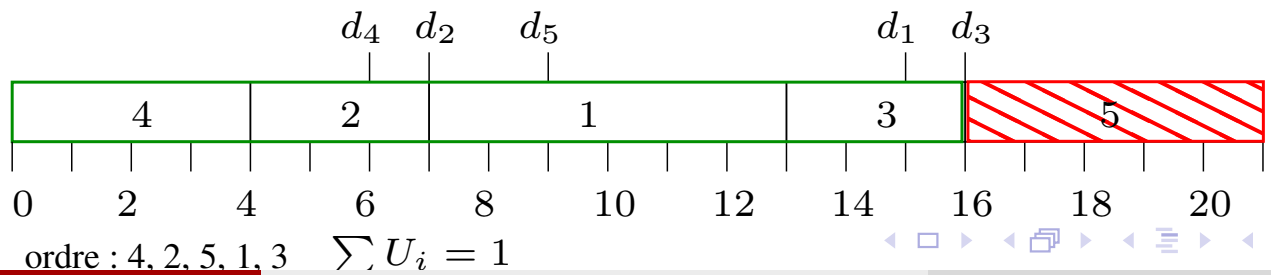
Ordonnancer sur une machine un ensemble de tâches définies par des durées p_i, des dates de livraison d_i, en minimisant le nombre de tâches en retard $\sum_{i \in T} U_i$.

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

Algorithme

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



EXERCICE

Exercice 4

Soit l'instance de problème à une machine :

i	p_i	w_i	d_i
1	6	4	14
2	3	3	8
3	7	5	17
4	4	5	7
5	5	1	10

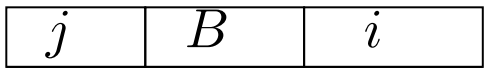
Appliquer l'algorithme de minimisation du plus grand retard ($L_{\max}$), celui de la somme pondérée des dates de fin ($\sum w_i C_i$) et celui de la somme des retards ($\sum U_i$). Pour chaque algorithme vous comparerez les valeurs des trois critères.

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

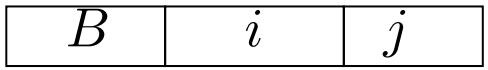
Le problème est NP-difficile (Karp, 1972). Le problème préemptif est donc également NP-difficile.

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 $F_j(t)$

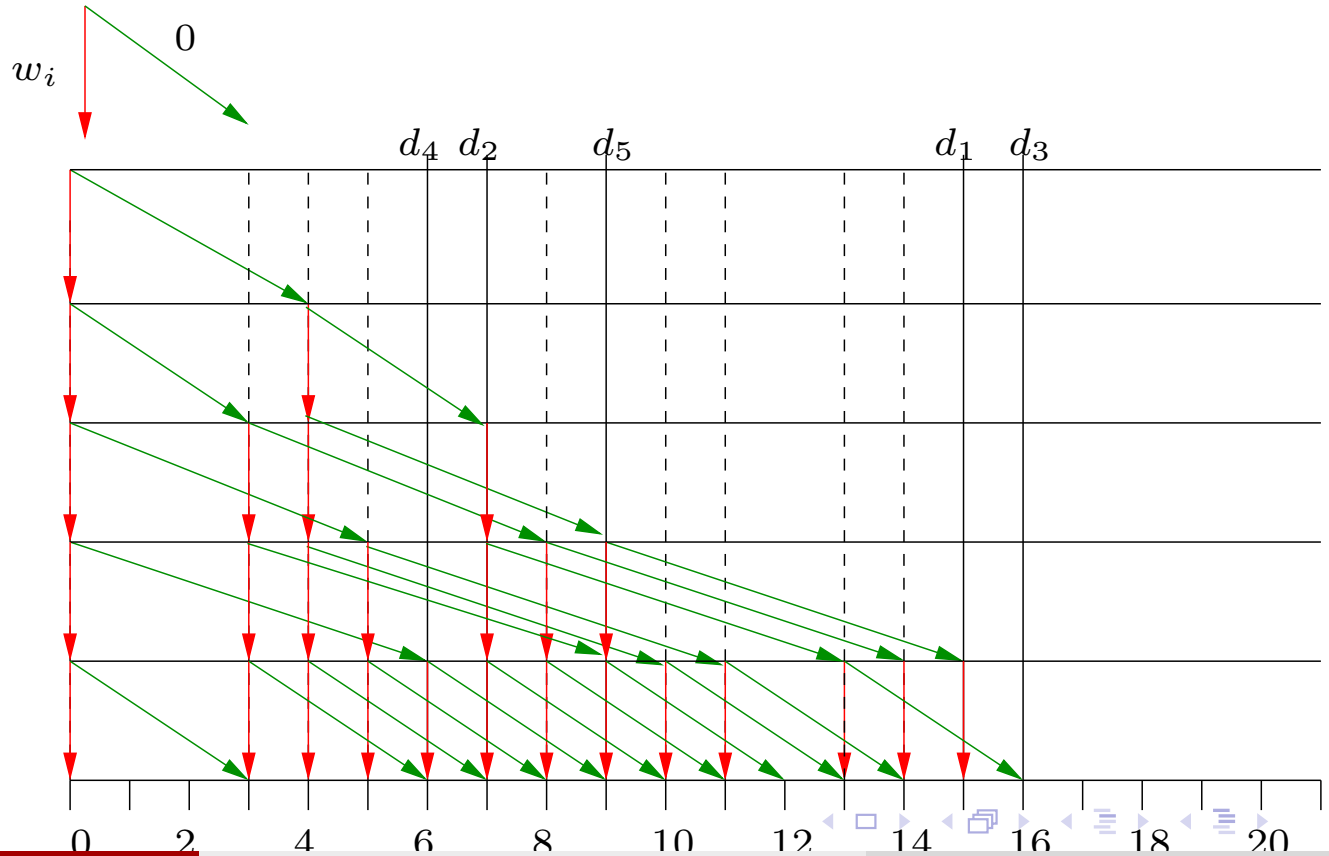
- Si $0 \leq t \leq d_i$ et $j \geq 1$ alors soit j est à l'heure à t et donc elle peut être placée à la fin de la séquence, soit j est en retard donc $F_j(t) = \min(F_{j-1}(t - p_j), F_{j-1}(t) + w_j)$.
- Si $t > d_j$ et $j \geq 1$ alors $F_j(t) = F_j(d_j)$ car toutes les tâches de $1, \ldots, j$ se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



EXERCICE

Exercice 5

Soit l'instance de problème à une machine de l'exercice 4. Appliquer l'algorithme de minimisation de la somme pondérée du nombre de tâches en retard. Comparer les valeurs des différents critères avec ceux des algorithmes de l'exercice 4.

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.

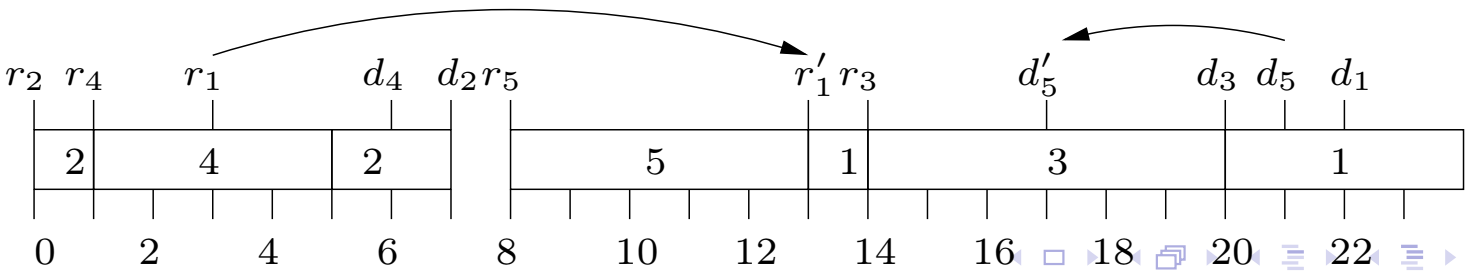
Plus grand retard avec préemption dates de lancement et précédences $1|prec, pmtn, r_i|L_{\max}$ (2/2)

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 livraison 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



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

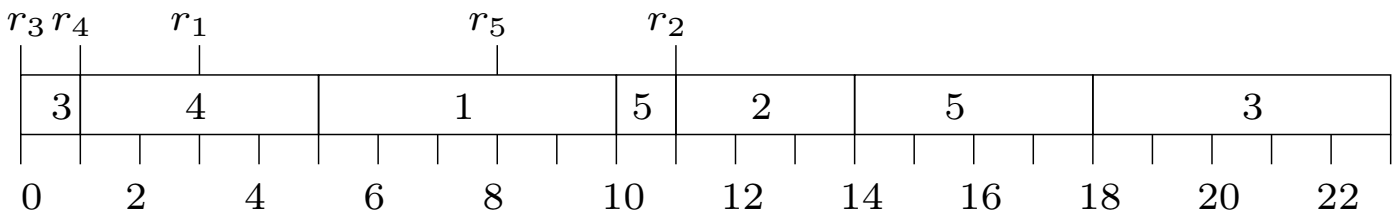
Définition

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)

A 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



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

EXERCICE

Exercice 6

Soit l'instance de problème à une machine :

i	p_i	r_i	d_i
1	6	3	14
2	3	11	8
3	7	0	17
4	4	1	7
5	5	8	10

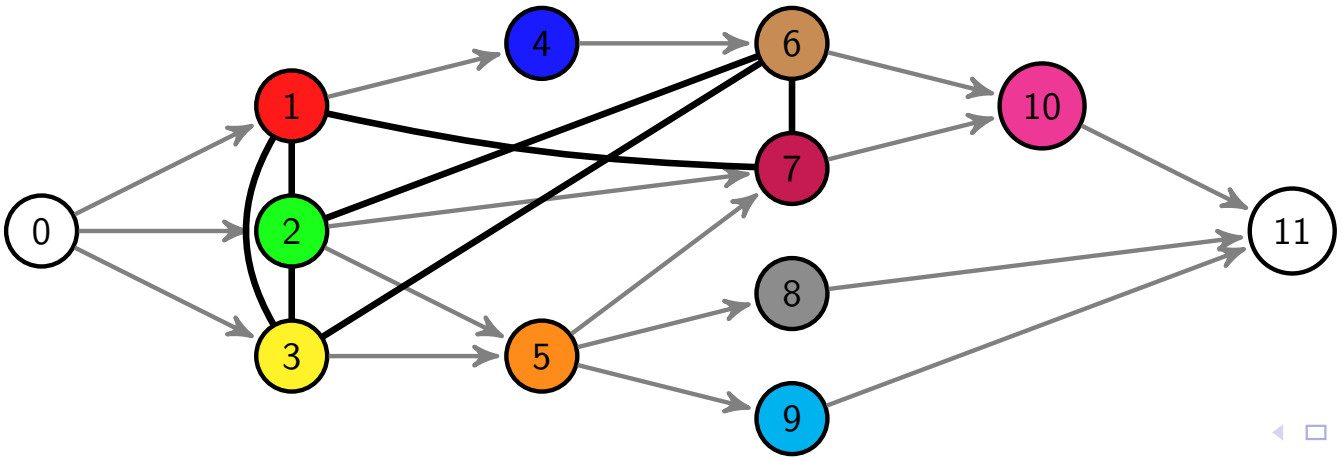
Résoudre $1|r_i, pmtn| \sum C_i$ et $1|prec, pmtn, r_i|L_{\max}$. Pour chaque algorithme vous comparerez les valeurs des deux critères.

Les problèmes d'ordonnancement à ressources non partageables : le graphe disjonctif

- La contrainte disjonctive exprime le non parallélisme entre deux tâches

$$S_j \geq S_i + p_i \vee S_i \geq S_j + p_j$$

- En général les tâches utilise une ressource non partageable
- D ensemble des paires de tâches en disjonction.
- Extension du graphe conjonctif : ajout d'une arête $\{i, j\}$ pour chaque paire de disjonction $\{i, j\} \in D$.



$$D = \{\{1,2\},\{1,3\},\{2,3\}$$
$$\{2,6\},\{3,6\},\{1,7\},\{6,7\}\}$$

Le graphe disjonctif : propriété fondamentale

Définition

Un ordonnancement S est semi-actif si pour tout $\epsilon > 0$ et toute tâche $i \in \mathcal{J}$, l'ordonnancement S' tel que $S'_i = S_i - \epsilon$ et $S'_j = S_i$, $\forall j \in \mathcal{J} \setminus \{i\}$ est irréalisable.

Propriété

L'ensemble des ordonnancement semi-actif est dominant pour toute fonction objectif régulière (c'est à dire non décroissante en fonction des dates de fin des tâches (e.g $\max_{j \in \mathcal{J}} S_j + p_j$, $\sum_{j \in \mathcal{J}} S_j + p_j$, ...)

Théorème

Il existe une bijection entre l'ensemble des ordonnancements semi-actifs et l'ensemble des orientations acycliques du graphe disjonctif

Problème disjonctif de grande taille : plans de contacts pour Galileo

Formulation du problème

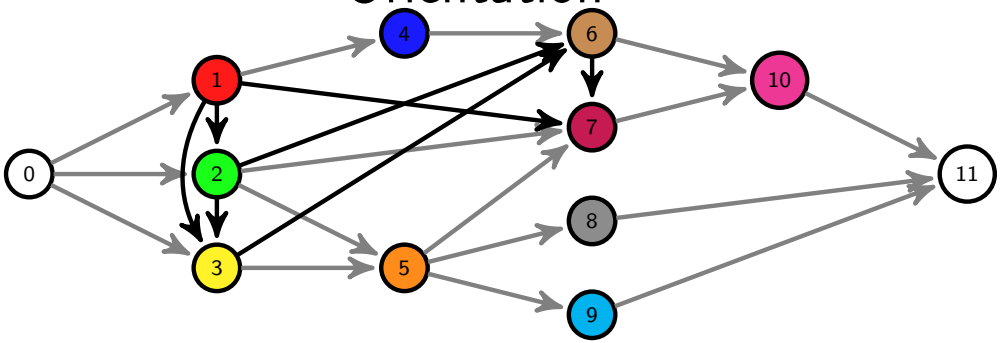
Trouver une orientation acyclique Π du graphe disjonctif qui minimise la longueur du plus long chemin entre 0 et $n+1$ (NP-difficile au sens fort). Équivaut à résoudre ce programme linéaire en variables mixtes à temps continu :

$$
\begin{aligned}
\min\ & S_{n+1} & \\
\text{s.c. } & S_j \geq S_i + l_{i,j} & \forall (i,j) \in E \\
& S_j \geq S_i + p_i + (1 - x_{ij}) M_{ij} & \forall \{i,j\} \in D \\
& x_{ij} + x_{ji} = 1 & \forall \{i,j\} \in D \\
& S_i \geq 0 & \forall i \in V \\
& x_{i,j} \in \{0,1\} & \forall \{i,j\} \in D
\end{aligned}
$$

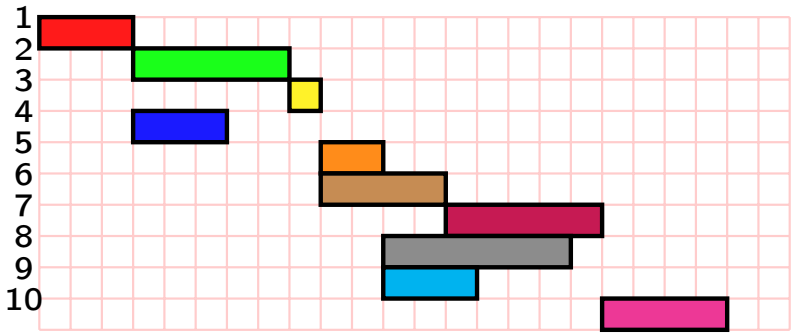
avec $M_{ij} - p_i$ une borne supérieure valide de $S_i - S_j$

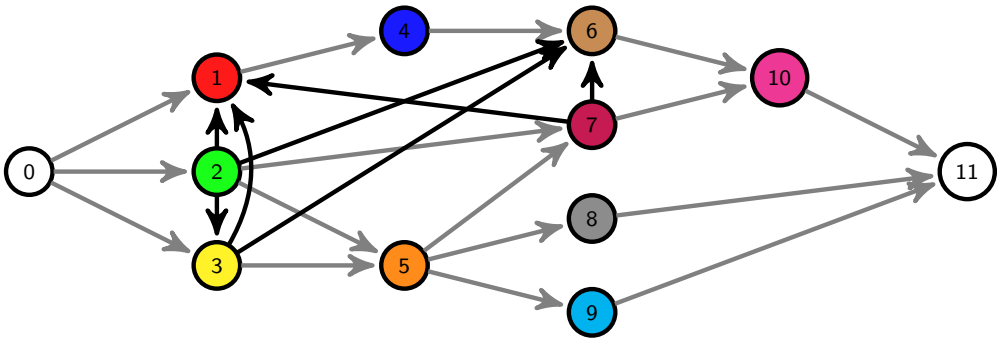
Exemples d'orientations réalisables et irréalisables

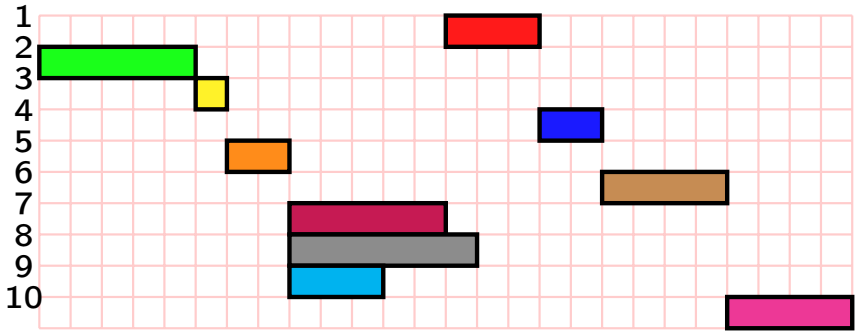
Orientation

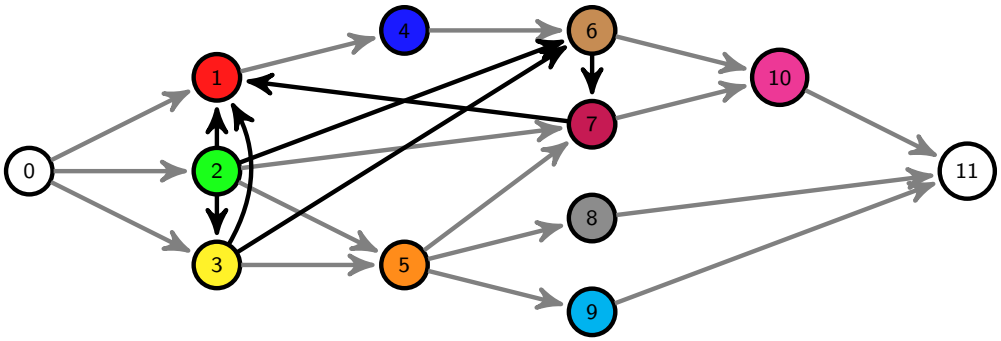


Ordonnancement au plus tôt





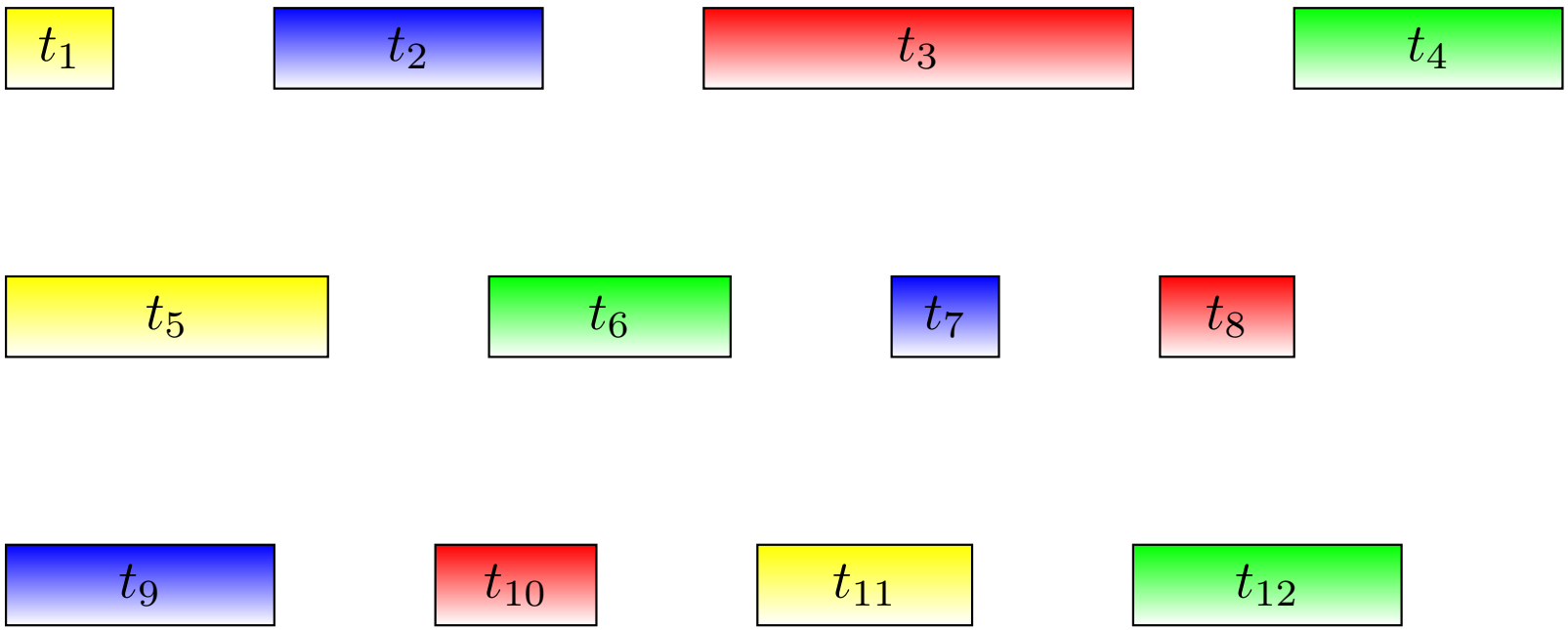




IRRÉALISABLE

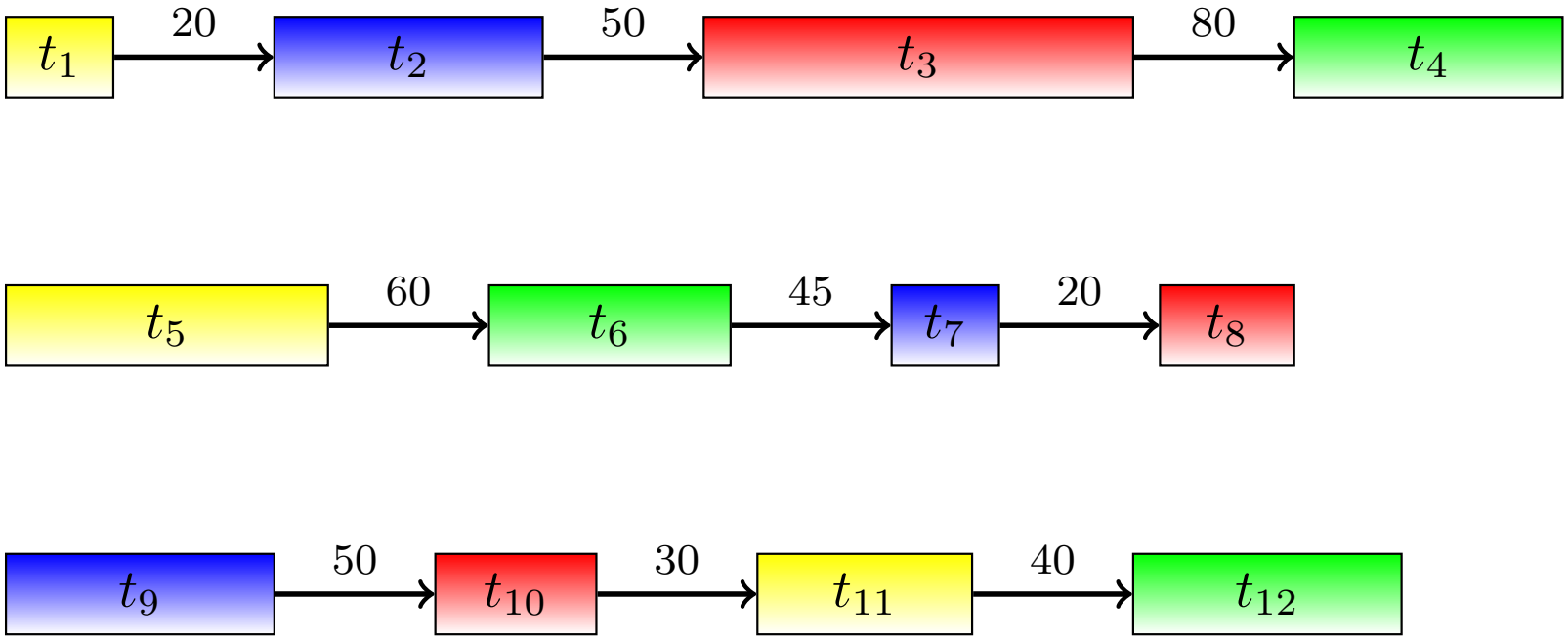
Exemple classique : le problème du Job-shop

N travaux décomposés en tâches, m machines $(N = 3, m = 4)$



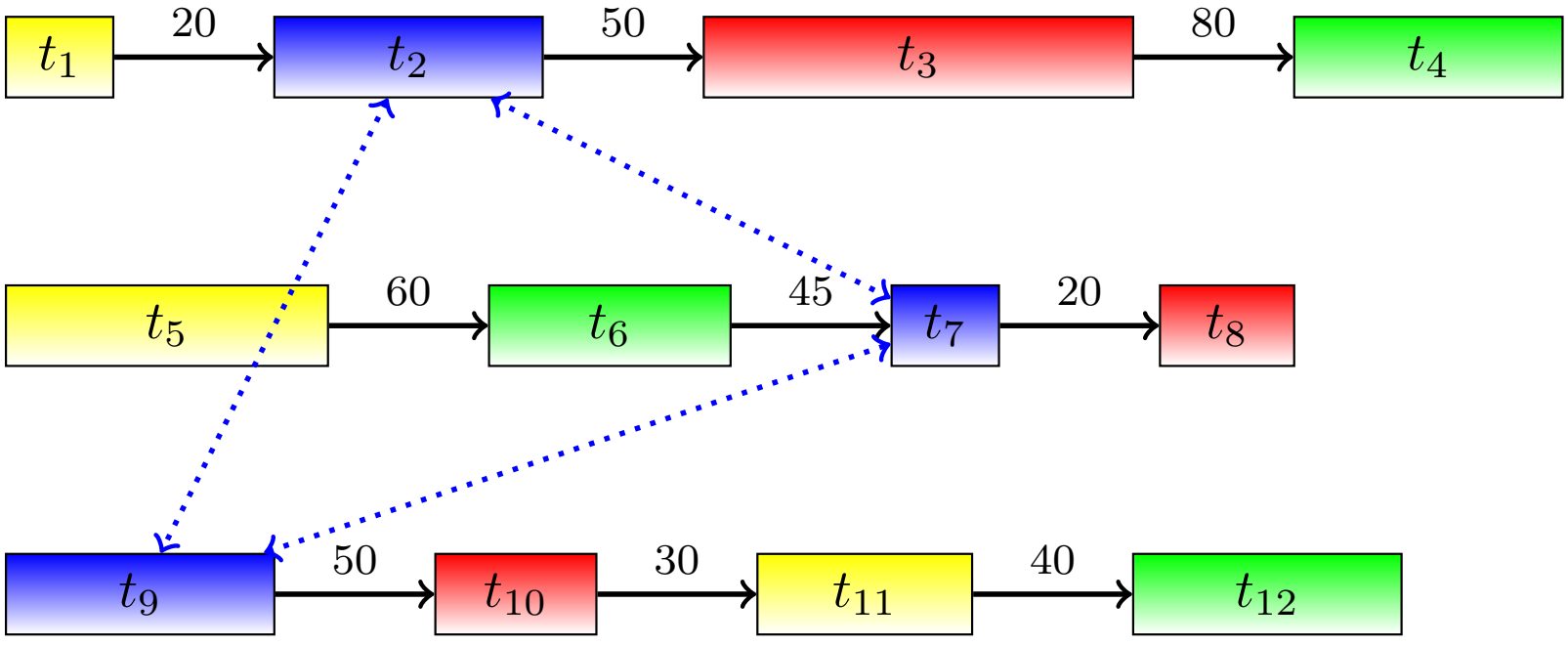
Exemple classique : le problème du Job-shop

N travaux décomposés en tâches, m machines $(N = 3, m = 4)$



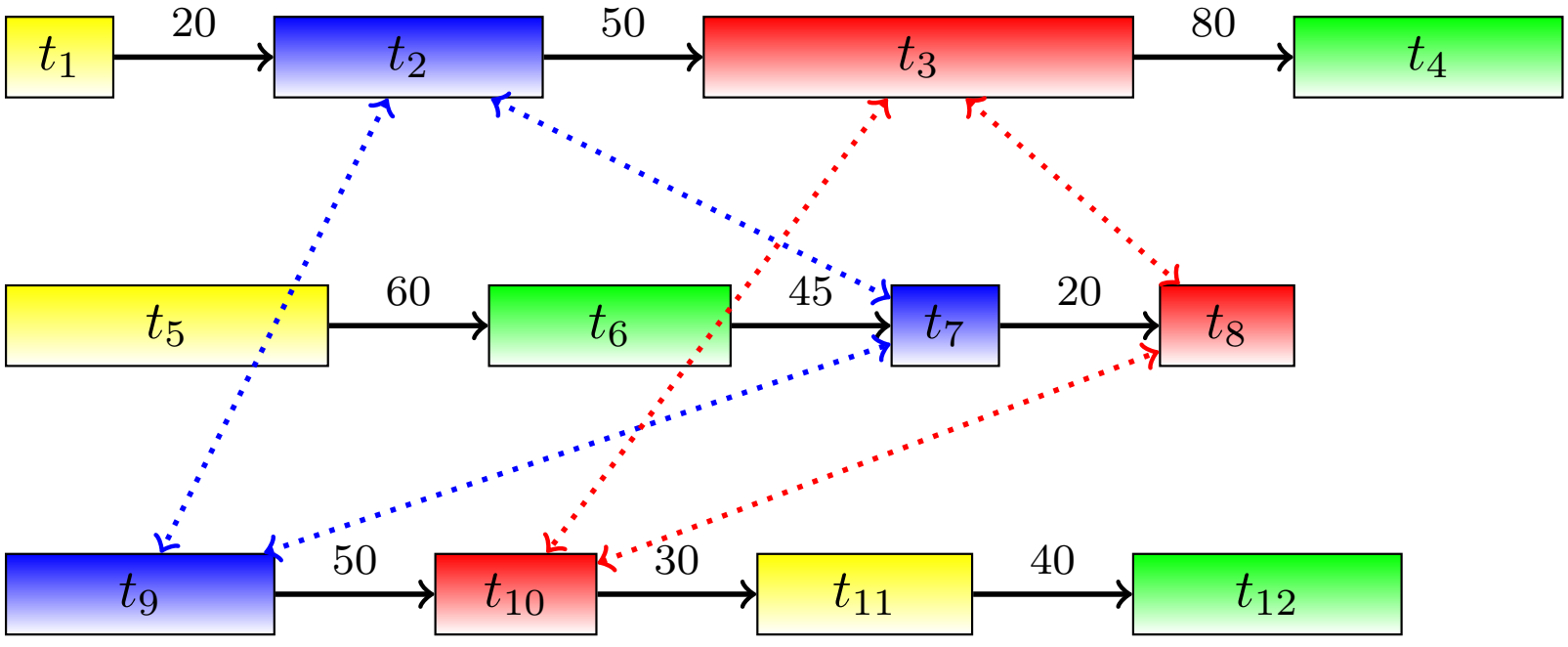
Exemple classique : le problème du Job-shop

N travaux décomposés en tâches, m machines $(N = 3, m = 4)$



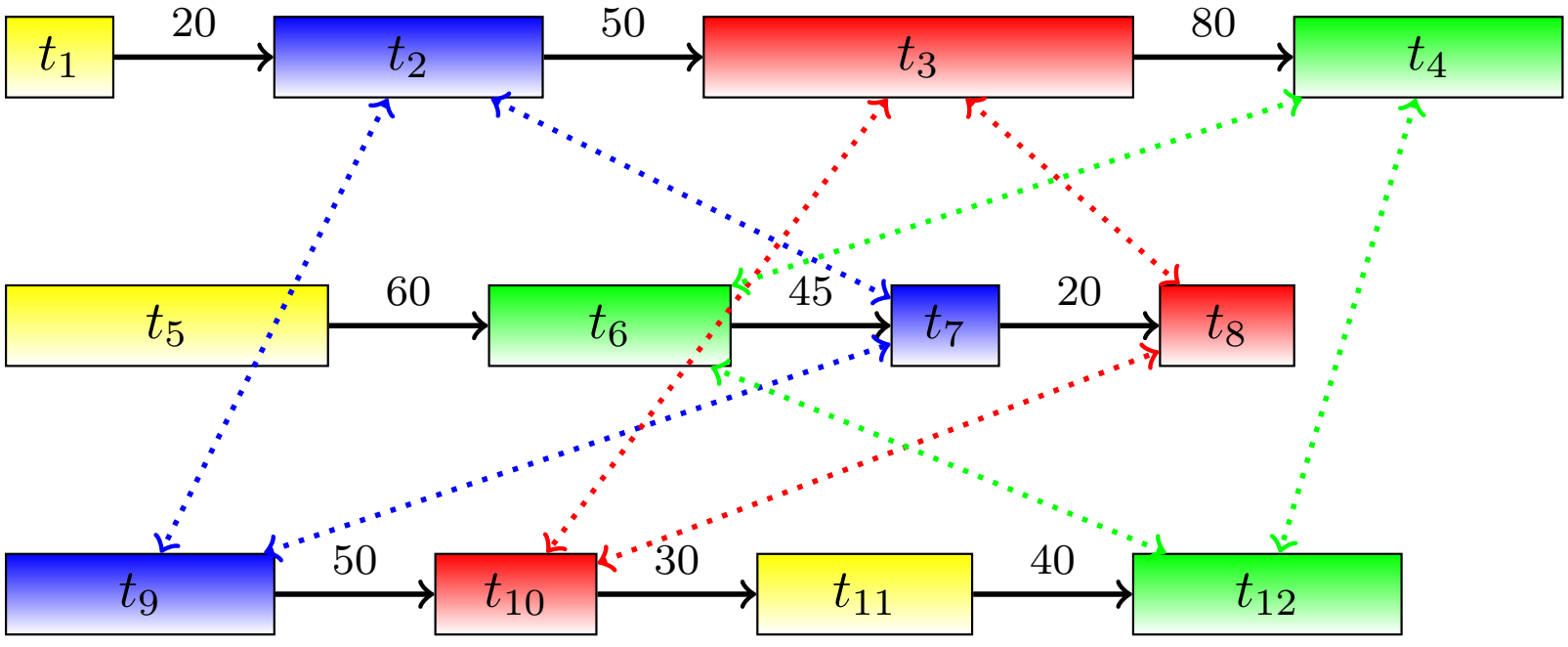
Exemple classique : le problème du Job-shop

N travaux décomposés en tâches, m machines $(N = 3, m = 4)$



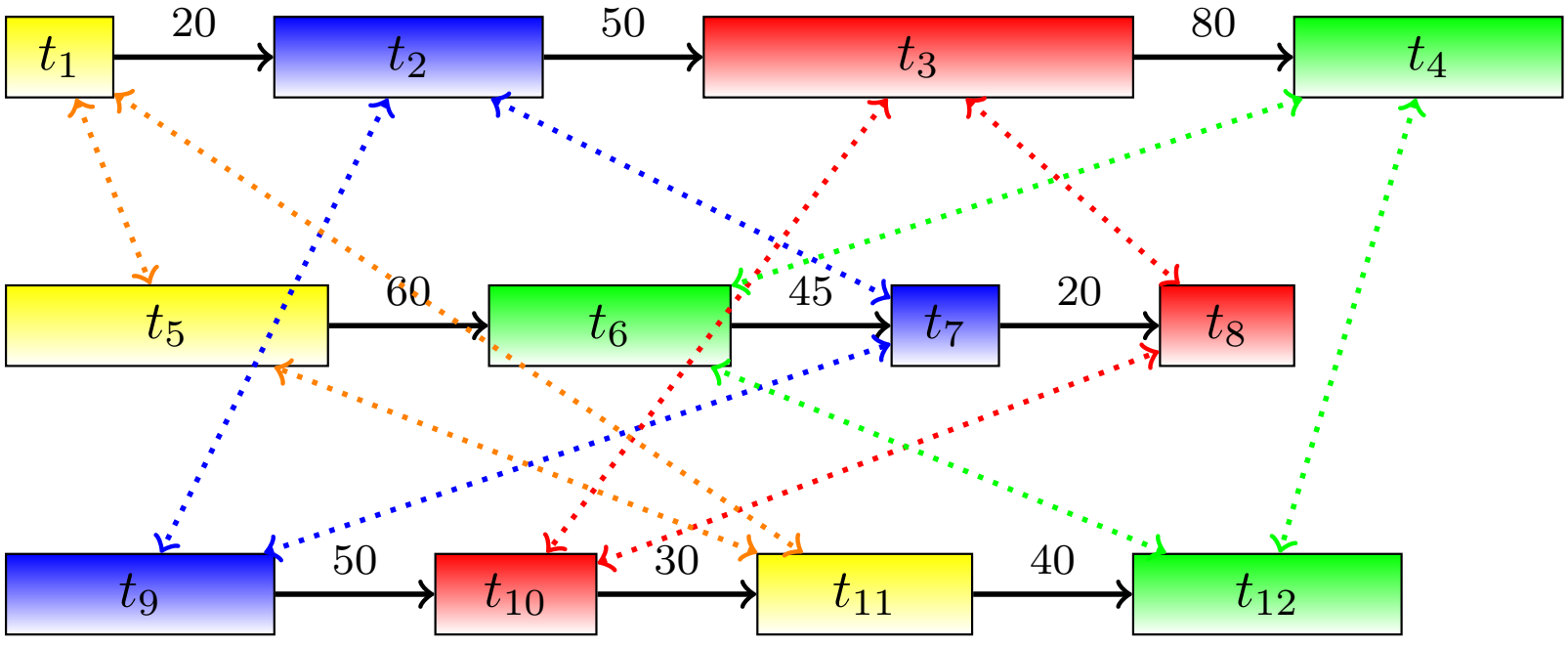
Exemple classique : le problème du Job-shop

N travaux décomposés en tâches, m machines $(N = 3, m = 4)$



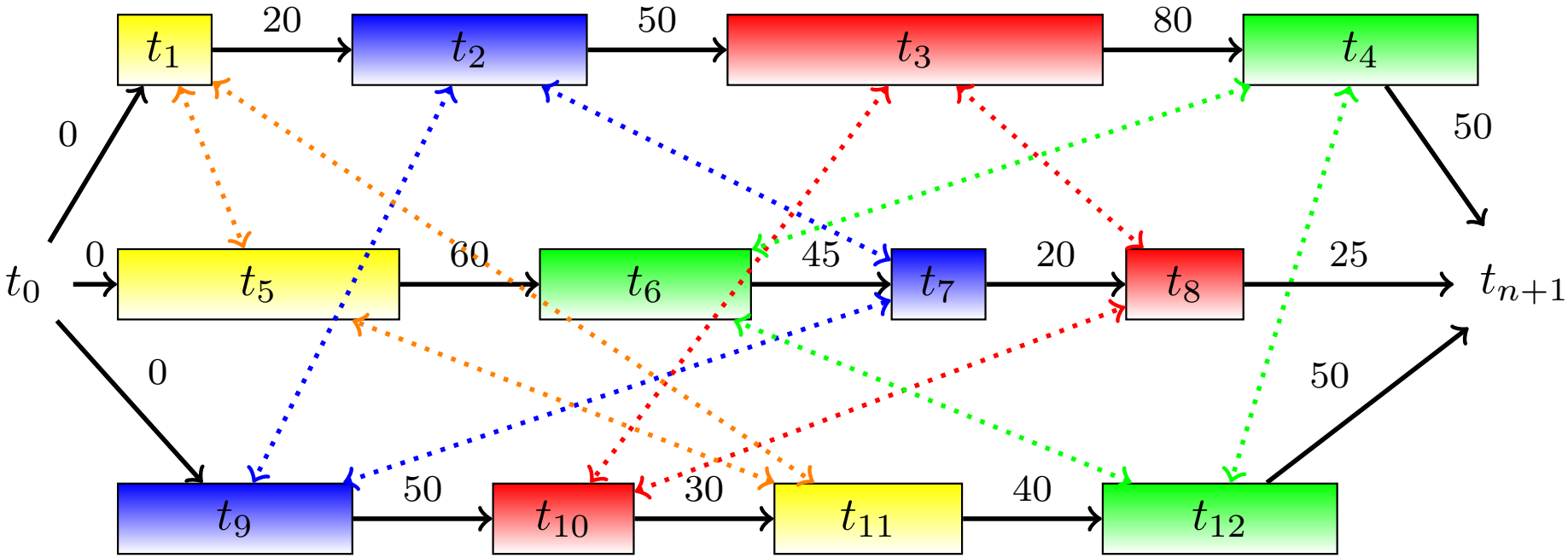
Exemple classique : le problème du Job-shop

N travaux décomposés en tâches, m machines $(N = 3, m = 4)$



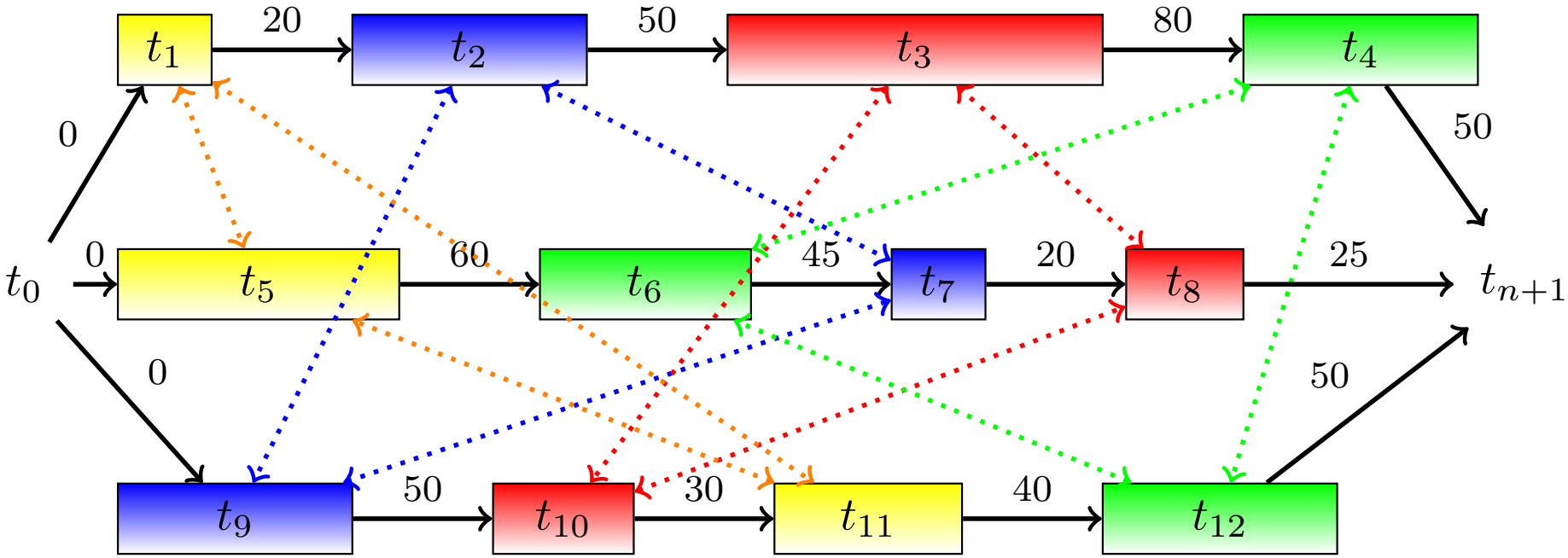
Exemple classique : le problème du Job-shop

N travaux décomposés en tâches, m machines $(N = 3, m = 4)$



Exemple classique : le problème du Job-shop

n travaux décomposés en tâches, m machines $(n = 3, m = 4)$



$\mathcal{J}_k$: ensemble des tâches exécutées sur machine k : clique maximale du graphe des disjonctions.

Utilisation de solveurs dédiés à l'ordonnancement

Le problème est fortement NP-difficile. Les industriels utilisent souvent des solveurs commerciaux pour les résoudre. Exemple : CP Optimizer (IBM)

```
# Create variables
for task in instance.tasks:
    interval_vars[task] = interval_var(start = (0, max_time), end = (0, max_time), size = task.length, name = 'interval'+str(task.name))

# Precedence and blocking constraints
for task in instance.tasks:
    if task.next_task:
        model.add(start_of(interval_vars[task.next_task]) >= end_of(interval_vars[task]))

# No overlap constraints
for machine in instance.machines:
    machine_sequence = sequence_var([interval_vars[task] for task in machine.tasks])
    model.add(no_overlap(machine_sequence))

# Minimize the makespan
obj_var = integer_var(0, max_time, 'makespan')
for task in instance.tasks:
    model.add(obj_var >= end_of(interval_vars[task]))
model.minimize(obj_var)

# Define solver and solve
sol = model.solve(TimeLimit= time_limit, Workers = threads)
```

Construire une solution réalisable l'algorithme de Giffler-Thompson

1. Inititialiser l'ensemble des tâches non ordonnancées Q avec les tâches sans prédécesseurs et les disponibilités des machines D_k à 0 pour toute machine k.
2. Mettre à jour les dates de début ES_i et de fin au plus tôt EF_i des tâches de Q de telle sorte qu'une tâche $i \in Q$ ne peut démarrer avant D_{m_i}.
3. Choisir la tâche j de plus petit EF_i parmi les tâches $i \in Q$.
4. Déterminer l'ensemble C des tâches en conflit avec j telles que $ES_i < EF_j$
5. Choisir une tâche i^* de de C avec une règle de priorité
6. Ordonnancer i^* à la date ES_{i^*}, enlever i^* de Q et ajouter son successeur j^* à Q et faire $ES_{j^*} \leftarrow EF_{i^*}$.
7. Si Q est vide STOP sinon revenir à l'étape 2.

EXERCICE 7

Soit un atelier réalisant 3 types de papier peint différents. Chaque type de papier est fabriqué sous forme d'un rouleau de papier continu qui passe sur plusieurs machines (chacune imprimant une couleur différente) L'ordre de passage (et la durée) sur les machines dépend du type de papier.

- Type 1 : Machine 1 (bleu, 45) puis Machine 3 (jaune, 15)
- Type 2 : Machine 2 (vert, 10), Machine 1 (bleu, 20) puis Machine 3 (jaune, 34)
- Type 3 : Machine 2 (vert, 17) puis Machine 3 (jaune, 28)

Une machine ne peut traiter qu'un seul type de papier à la fois et qu'un type de papier ne peut passer que sur une machine à la fois On souhaite réaliser ces 3 types de produits avec une durée minimale.

Questions

- Proposer une modélisation du problème sous la forme d'un graphe disjonctif
- Trouver une solution en appliquant l'algorithme de Giffler-Thompson avec la règle de priorité de la plus grande durée restante à réaliser dans le travail après la fin de l'opération.
- Modéliser cette solution sous la forme d'un graphe conjonctif
- Donner le chemin critique.