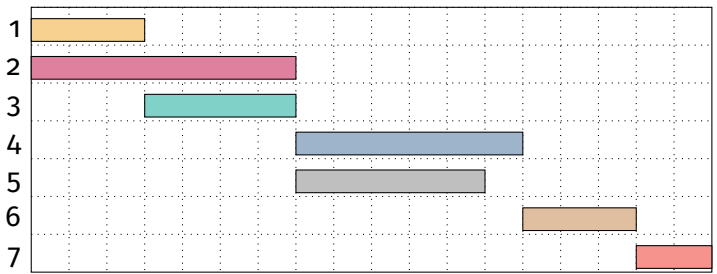
RO03 : Recherche opérationnelle

Problèmes d'ordonnancement

Antoine Jouglet, David Savourey



Introduction

Ordonnancement

Définition

Un problème d'ordonnancement consiste à **déterminer des dates d'exécution** pour un ensemble d'**activités** (ou tâches, ou jobs) qui utilisent un ensemble de **ressources** dont la capacité est limitée.

- système de production ;
- transport ;
- conception d'emplois du temps ;
- *etc.*

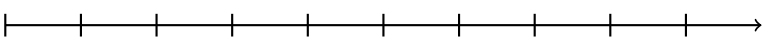
Les jobs

Un ensemble de n jobs $N = \{1, 2, \ldots, n\}$:

- durée p_i,
- date de disponibilité r_i (sinon par défaut $r_i = 0$),
- date échue d_i (si on minimise le retard),
- poids w_i (sinon par défaut $w_i = 1$).

Pour chaque job i ordonnancé :

- date de début t_i,
- date de fin C_i,
- retard $T_i = \max(0, C_i - d_i)$.



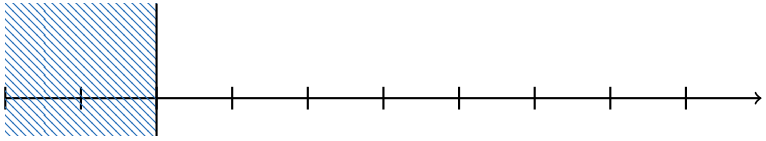
Les jobs

Un ensemble de n jobs $N = \{1, 2, \ldots, n\}$:

- durée p_i,
- date de disponibilité r_i (sinon par défaut $r_i = 0$),
- date échue d_i (si on minimise le retard),
- poids w_i (sinon par défaut $w_i = 1$).

Pour chaque job i ordonnancé :

- date de début t_i,
- date de fin C_i,
- retard $T_i = \max(0, C_i - d_i)$.



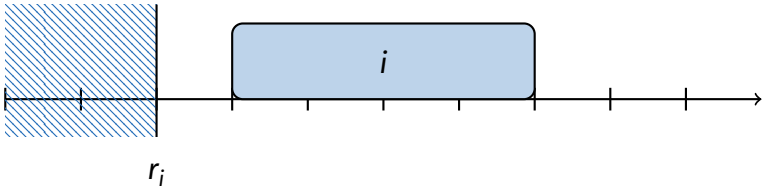
Les jobs

Un ensemble de n jobs $N = \{1, 2, \ldots, n\}$:

- durée p_i,
- date de disponibilité r_i (sinon par défaut $r_i = 0$),
- date échue d_i (si on minimise le retard),
- poids w_i (sinon par défaut $w_i = 1$).

Pour chaque job i ordonnancé :

- date de début t_i,
- date de fin C_i,
- retard $T_i = \max(0, C_i - d_i)$.



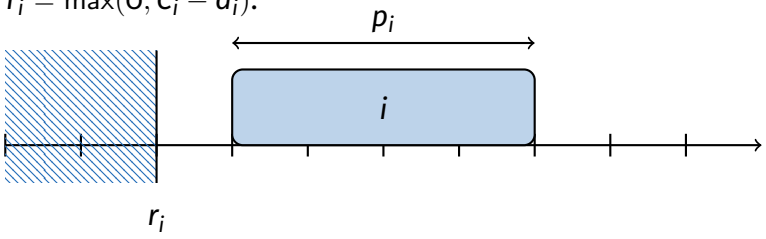
Les jobs

Un ensemble de n jobs $N = \{1, 2, \ldots, n\}$:

- durée p_i,
- date de disponibilité r_i (sinon par défaut $r_i = 0$),
- date échue d_i (si on minimise le retard),
- poids w_i (sinon par défaut $w_i = 1$).

Pour chaque job i ordonnancé :

- date de début t_i,
- date de fin C_i,
- retard $T_i = \max(0, C_i - d_i)$.



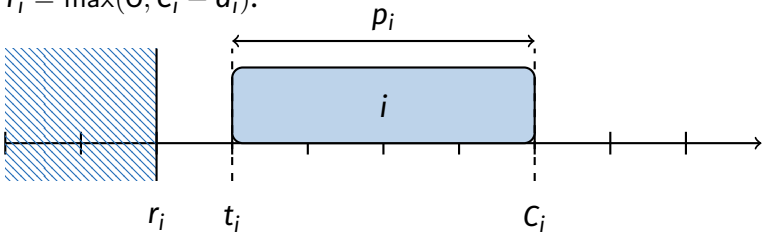
Les jobs

Un ensemble de n jobs $N = \{1, 2, \ldots, n\}$:

- durée p_i,
- date de disponibilité r_i (sinon par défaut $r_i = 0$),
- date échue d_i (si on minimise le retard),
- poids w_i (sinon par défaut $w_i = 1$).

Pour chaque job i ordonnancé :

- date de début t_i,
- date de fin C_i,
- retard $T_i = \max(0, C_i - d_i)$.



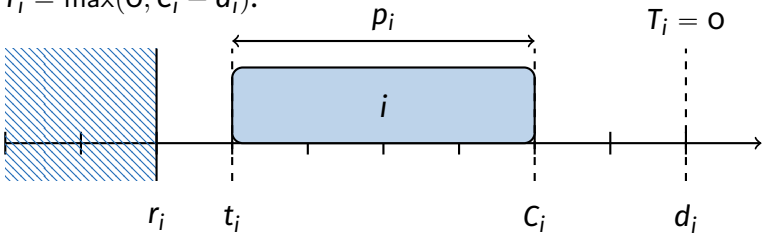
Les jobs

Un ensemble de n jobs $N = \{1, 2, \ldots, n\}$:

- durée p_i,
- date de disponibilité r_i (sinon par défaut $r_i = 0$),
- date échue d_i (si on minimise le retard),
- poids w_i (sinon par défaut $w_i = 1$).

Pour chaque job i ordonnancé :

- date de début t_i,
- date de fin C_i,
- retard $T_i = \max(0, C_i - d_i)$.



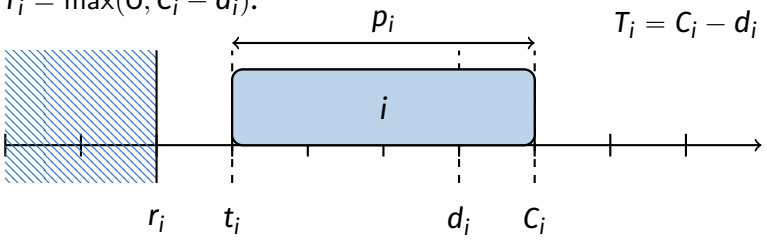
Les jobs

Un ensemble de n jobs $N = \{1, 2, \ldots, n\}$:

- durée p_i,
- date de disponibilité r_i (sinon par défaut $r_i = 0$),
- date échue d_i (si on minimise le retard),
- poids w_i (sinon par défaut $w_i = 1$).

Pour chaque job i ordonnancé :

- date de début t_i,
- date de fin C_i,
- retard $T_i = \max(0, C_i - d_i)$.

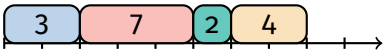


Les ressources

différents types de ressource

Les ressources

différents types de ressource

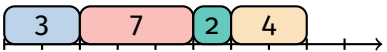


préemption interdite

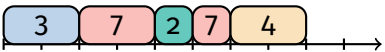
préemption autorisée

Les ressources

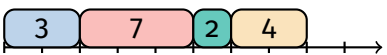
différents types de ressource



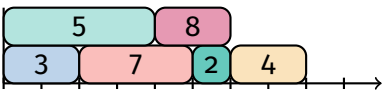
préemption interdite



préemption autorisée



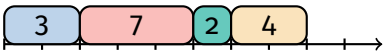
une machine



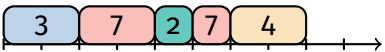
machines parallèles

Les ressources

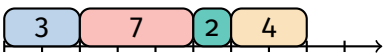
différents types de ressource



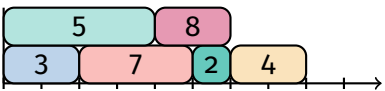
préemption interdite



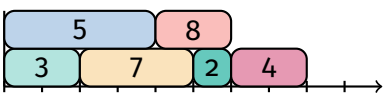
préemption autorisée



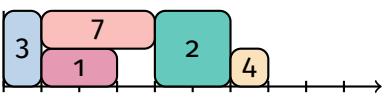
une machine



machines parallèles



disjonctif



cumulatif

Précédences

Les jobs peuvent être soumis à des relations de précédences, données sous forme d'un graphe orienté sans circuit (*DAG* pour *Direct Acyclic Graph*).

Plusieurs types de précédences peuvent exister :

- le job i doit débuter avant le job j
- **le job i doit être terminé avant le début du job j**
- le job i doit terminer avant le job j

Critères d'optimisation

Définition

Quand on cherche un ordonnancement qui respecte l'ensemble des contraintes, on parle d'**ordonnancement réalisable**. C'est un **problème de faisabilité**.

Définition

On peut se donner un **critère** qui permet d'attribuer un score à chaque ordonnancement réalisable. Trouver un ordonnancement qui maximise ou minimise ce critère est un **problème d'optimisation**.

Exemples de critères courants en ordonnancement

- la date de fin du projet $C_{max} = \max_i C_i$ (appelé *makespan*)
- la somme des dates de fin $\sum_i C_i$
- la somme des dates de fin pondérées $\sum_i w_i C_i$
- la somme des retards $\sum_i T_i$
- *etc.*

Problème central

Un exemple de problème

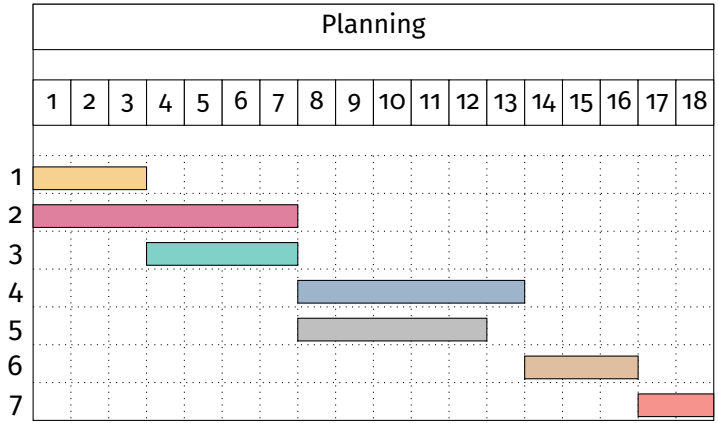
Tâche	Durée	Précédence(s)
1	3	
2	7	
3	4	précédée par 1
4	6	précédée par 1 et 2
5	5	précédée par 3
6	3	précédée par 3 et 4
7	2	précédée par 6

Objectifs :

- Trouver l'ordonnancement qui **minimise la date de fin du projet**, avec un nombre non limité de machines et en respectant les précédences ;
- Identifier les **tâches critiques**, *i.e.* les tâches dont le retard entraîne nécessairement un retard du projet.

Diagramme de Gantt

Tâche	Durée	Préc.
1	3	
2	7	
3	4	1→3
4	6	1,2→4
5	5	3→5
6	3	3,4→6
7	2	6→7



le temps en abscisse, les tâches en ordonnée

Graphes conjonctifs et potentiels

Graphe conjonctif

On construit un **graphe qui modélise le problème d'ordonnancement**, avec :

- sommet = étape ou évènement
- arc = contrainte de précédence
- ordonnancement = ensemble de date associée à chaque sommet

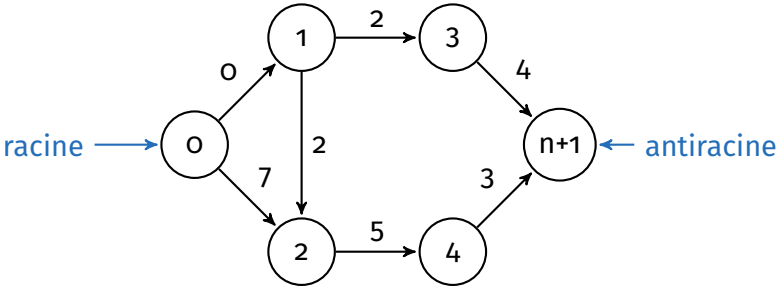
Graphe conjonctif : définition

Définition

Un **graphe conjonctif** est un graphe valué $G = (X, U, v)$ ayant une racine 0 et une antiracine $n + 1$ tel que :

- il existe un chemin de valeur positive entre la racine et tout autre sommet
- il existe un chemin de valeur positive entre tout autre sommet et l'antiracine

Graphe conjonctif : exemple



Ensemble de potentiels

Définition

Un **ensemble de potentiels** sur un graphe conjonctif $G = (X, U, v)$ est une application $t : X \to \mathbb{R}$ telle que :

- $t_0 = 0$;
- $\forall (i, j) \in U, t_i + v_{ij} \leqslant t_j$.

Remarque

Un ensemble de potentiels est un ensemble de dates de début d'exécution pour chaque tâche, correspondant à un ordonnancement réalisable.

Existence d'un ensemble de potentiels

Théorème

Il existe un ensemble de potentiels pour un graphe conjonctif si et seulement si **ce graphe ne contient pas de circuit de valeur strictement positive**.

$\Rightarrow$ S'il existe un circuit strictement positif, alors il y aura nécessairement un arc de ce circuit qui ne respectera pas $t_i + v_{ij} \leqslant t_j$.

$\Leftarrow$ S'il n'existe pas de circuit strictement positif, on sait qu'il existe des chemins maximaux. L'ensemble des valeurs de ces plus longs chemins sont des potentiels.

Chemins maximaux

Notation

On note $l(i,j)$ la valeur maximale d'un chemin allant de i à j.

Remarque

On obtient les chemins maximaux avec les algorithmes de plus courts chemins en remplaçant les min par des max.

Potentiels au plus tôt

Théorème

L'ensemble $R = \{r_i = l(o, i), i \in X\}$ est un ensemble de potentiels. On appelle R l'ensemble des potentiels **câlés à gauche** ou **au plus tôt**.

On note $t^* = r_{n+1}$ le potentiel au plus tôt de l'antiracine. C'est la **date de fin au plus tôt du projet**.

Potentiels au plus tôt : propriété

Théorème

Pour tout ensemble de potentiels $T = \{t_i, i \in X\}$, on a $\forall i \in X, t_i \geqslant r_i$.

En particulier, on a $t_{n+1} \geqslant t^* = r_{n+1}$.

Preuve : On considère un chemin maximal C entre o et i. Pour tout arc (u, v) de C on a, $t_u + v_{uv} \leqslant t_v$. En sommant toutes ces inégalités sur les arcs de C, on obtient $t_o + v(C) \leqslant t_i$. Or $t_o = o$ et $v(C) = r_i$, donc $r_i \leqslant t_i$.

Potentiels au plus tard

Théorème

L'ensemble $F = \{f_i = l(0, n+1) - l(i, n+1), i \in X\}$ est un ensemble de potentiels. On appelle F l'ensemble des potentiels **câlés à droite** ou **au plus tard**.

Remarque

F correspond aux dates au plus tard auxquelles doivent démarrer les tâches pour que le projet termine au plus tôt.

Potentiels au plus tard : propriété

Théorème

Quelque soit l'ensemble de potentiels $T = \{t_i, i \in X\}$ tel que $t_{n+1} = t^*$, on a $\forall i \in X, t_i \leqslant f_i$.

Preuve : Supposons qu'il existe i tel que $t_i > f_i$. Alors $t_i > l(0, n+1) - l(i, n+1)$, donc $t_i + l(i, n+1) > l(0, n+1)$, d'où $t_{n+1} > t^*$. Donc le projet ne termine pas en t^*, ce qui est contradictoire.

Chemins critiques

Définition

On appelle **chemins critiques** les chemins maximaux entre 0 et $n+1$. Les tâches sur un chemin critique sont dites **tâches critiques**.

Si on retarde l'exécution d'une tâche critique, on retarde la date de fin du projet.

Les tâches critiques ont leur date au plus tôt égale à leur date au plus tard : elles sont de **marge nulle**.

Ordonnancement

Calculer un ordonnancement consiste à **assigner des dates de démarrage** aux tâches.

Tout ordonnancement réalisable est de **durée supérieure à** t^*.

On calcule en général **les dates au plus tôt** et **les dates au plus tard** en utilisant Bellman.

Tout ordonnancement qui est compris **entre les dates au plus tôt et les dates au plus tard** est **optimal**.

Méthodes de modélisation

2 méthodes fondées sur les graphes

Méthode potentiels tâches :

- française
- 1958
- construction paquebot France

Méthode PERT :

- américaine
- 1958
- construction fusée Polaris

Dans les deux cas :

- On cherchait à minimiser le temps de construction (et non le coût de construction);
- Ne prennent pas en compte les contraintes de ressources;
- Basées sur les algorithmes de recherche de chemins maximaux.

Méthode potentiels tâches

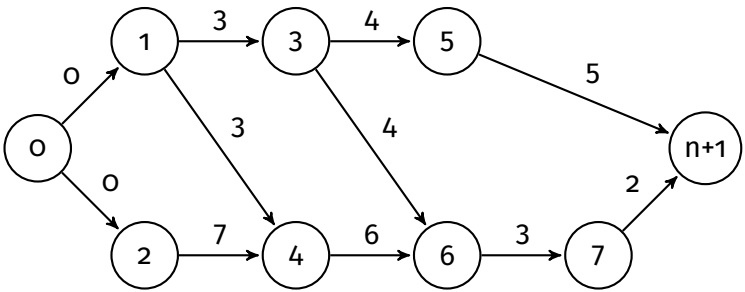
Méthode

On associe au problème d'ordonnancement le graphe conjonctif $G = (X, U, v)$ où :

- $X = \{1, 2, \ldots, n\} \cup \{0, n+1\}$
- Si la tâche i précède j, alors on ajoute un arc (i, j) avec $v_{ij} = p_i$
- Si la tâche i sans prédécesseur, on ajoute un arc $(0, i)$ avec $v_{0i} = 0$
- Si la tâche i sans successeur, on ajoute un arc $(i, n+1)$ avec $v_{in+1} = p_i$

Méthode potentiels tâches : exemple

Tâche	Durée	Préc.
1	3	
2	7	
3	4	1→3
4	6	1,2→4
5	5	3→5
6	3	3,4→6
7	2	6→7



Contraintes potentielles et graphe conjonctif

Contrainte de succession :

- ordo : i précède j, soit $t_i + p_i \leqslant t_j$
- graphe : arc (i, j) avec $v_{ij} = p_i$

Date de disponibilité :

- ordo : i doit commencer après la date r_i, soit $t_i \geqslant r_i$
- graphe : arc (o, i) avec $v_{\mathrm{o}i} = r_i$

Date échue :

- ordo : i doit terminer avant la date d_i, soit $t_i + p_i \leqslant d_i$
- graphe : arc (i, o) avec $v_{i\mathrm{o}} = p_i - d_i$ $(\leqslant \mathrm{o})$

Méthode PERT

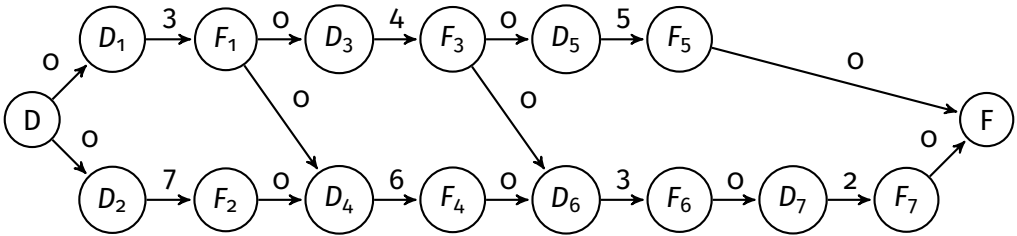
Méthode

On associe au problème d'ordonnancement le graphe conjonctif $G = (X, U, v)$ où :

- deux sommets D et F
- pour chaque tâche i, un sommet D_i et un sommet F_i
- pour chaque tâche i, un arc (D_i, F_i) avec $v_{D_iF_i} = p_i$
- si i précède j, un arc (F_i, D_j) avec $v_{F_iD_j} = 0$
- Si la tâche i sans prédécesseur, on ajoute un arc (D, D_i) avec $v_{DD_i} = 0$
- Si la tâche i sans successeur, on ajoute un arc (F_i, F) avec $v_{F_iF} = 0$

Méthode PERT : exemple

Tâche	Durée	Préc.
1	3	
2	7	
3	4	1→3
4	6	1,2→4
5	5	3→5
6	3	3,4→6
7	2	6→7



Graphe PERT simplifié

On **fusionne certains sommets** du graphe :

- sommets reliés par arcs nuls
- ne doit pas créer de nouveaux chemins

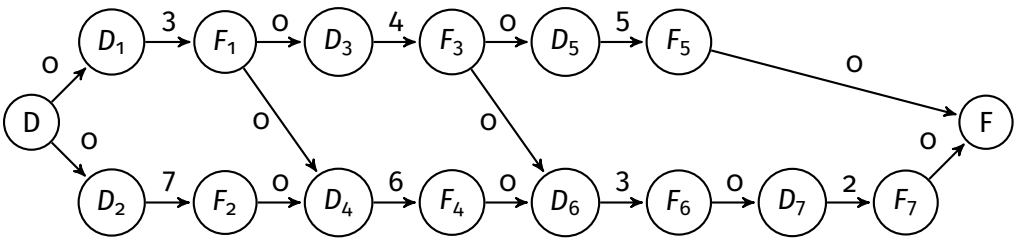
avantage :

- rend le graphe beaucoup plus lisible

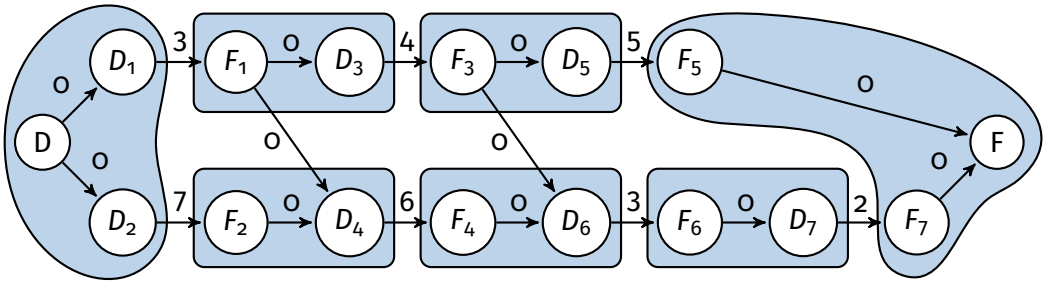
inconvénients :

- non automaticité de sa construction
- non unicité
- si on modifie une contrainte, il faut le reconstruire entièrement

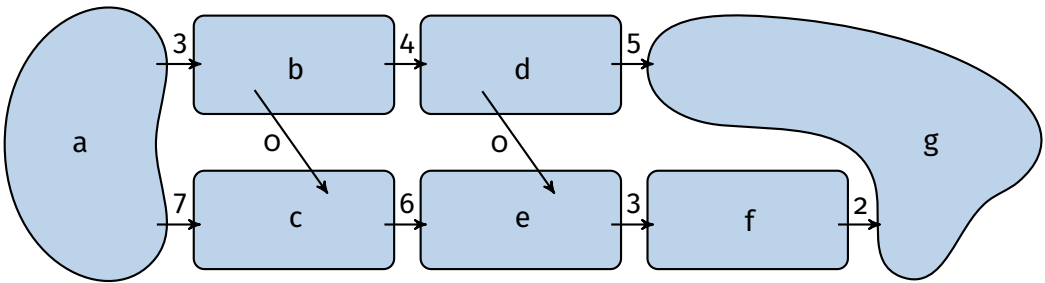
Graphe PERT simplifié



Graphe PERT simplifié



Graphe PERT simplifié



Graphe PERT simplifié