Systèmes à événements discrets et Ordonnancement : Théories, applications et comparaison aux approches de Recherche Opérationnelle

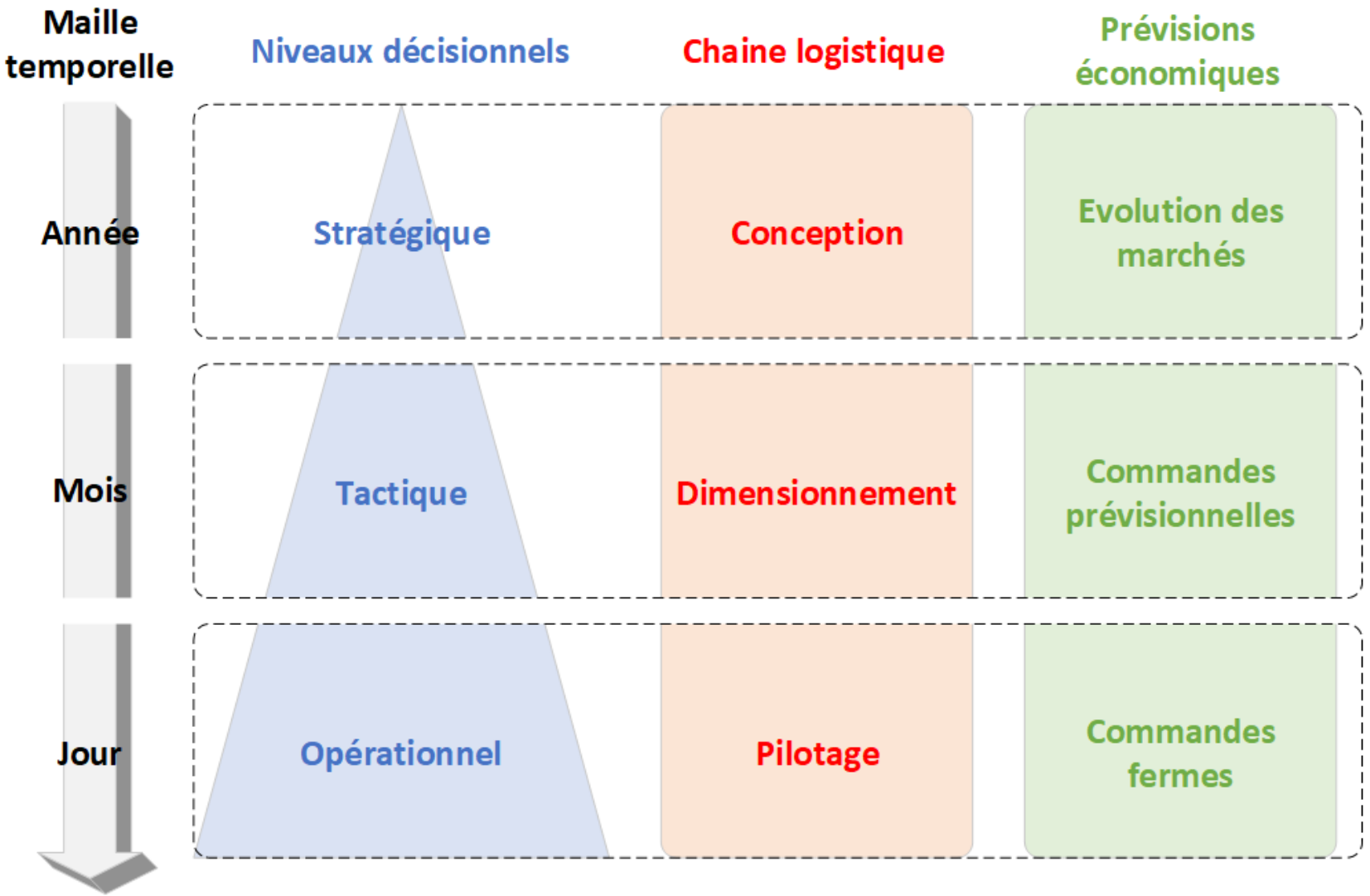
A. Aubry, S. Himmiche, D. Lemoine, P. Marange, S. Norre

Ecole MACS SED-Bermudes

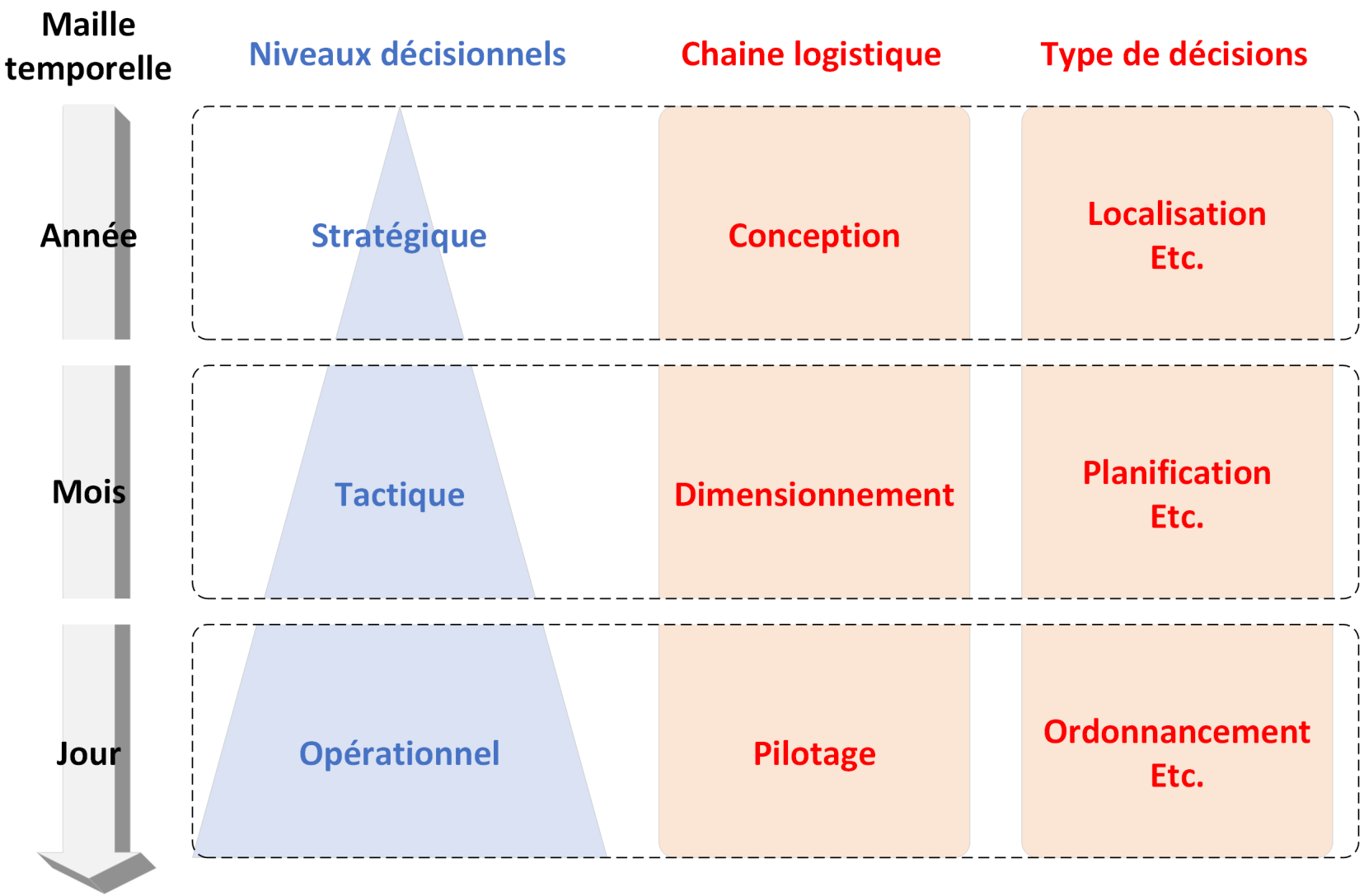
04 Juin 2019

Sommaire

Positionnement



Positionnement



Qu'est ce que l'ordonnancement ?

Une **tâche** est localisée dans le temps par une date de début et/ou de fin. Sa réalisation requiert une **durée** et nécessite des **ressources**.

Problème d'ordonnancement

un **problème d'ordonnancement** consiste à :
≪ programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution ≫
(Carlier et Chrétienne, 1988).

Qu'est ce que l'ordonnancement ?

Ressource

Une **ressource** est un moyen technique ou humain nécessaire à la réalisation d'une tâche. Lorsque celle-là est disponible en quantité limitée, on introduit alors la notion de « capacité ».

On retrouve deux types de ressource :

1. les ressources **renouvelables**,
2. les ressources **non renouvelables** ou **consommables**.

Contraintes

Les tâches peuvent être soumises à des contraintes :

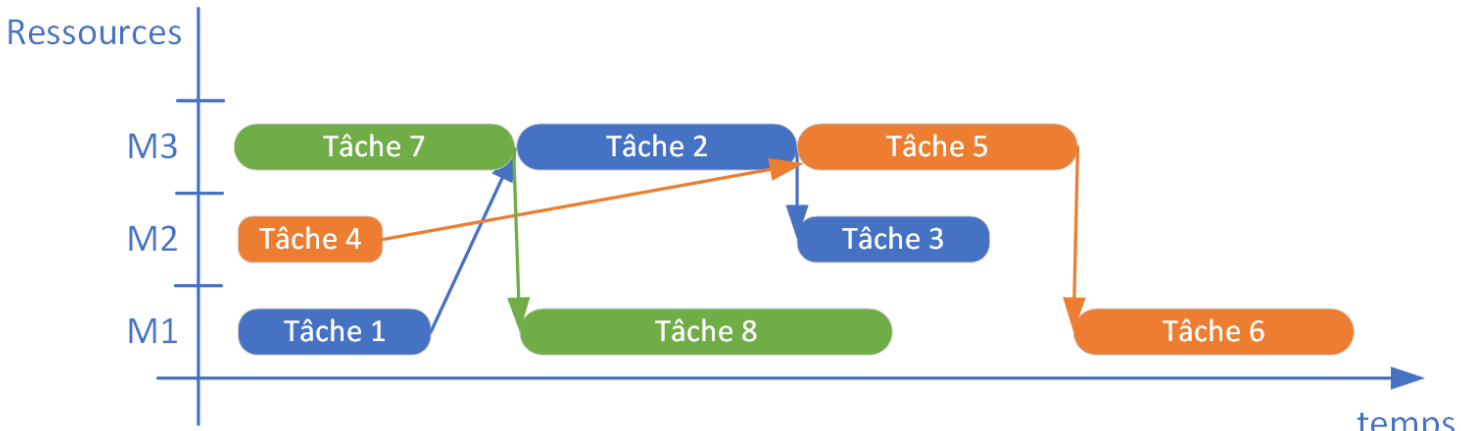
1. de **précédence**,
2. de **préemption** ou de **non préemption**.

Qu'est ce que l'ordonnancement ?

Ordonnancement

Un **ordonnancement** est une solution au problème d'ordonnancement, dans laquelle sont fixées les dates de début et de fin d'exécution des tâches et les ressources qui leurs sont allouées.

Un moyen commode de représenter un ordonnancement est un **Diagramme de Gantt** *(représentation temporelle de l'occupation des ressources)*



Qu'est ce que l'ordonnancement ?

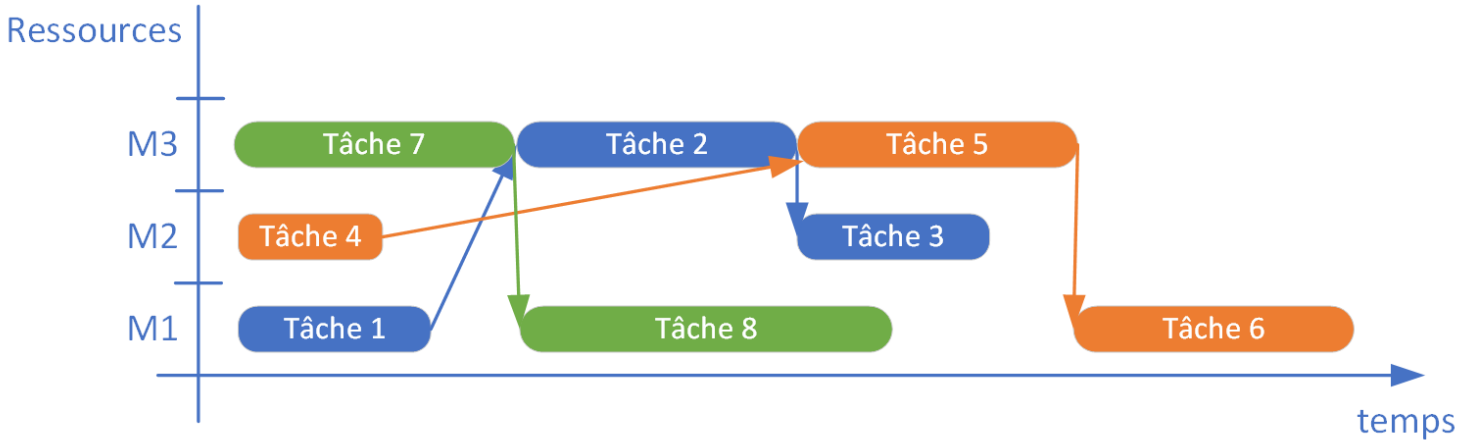
Pour un problème d'ordonnancement donné, il existe une infinité d'ordonnancements possibles. Il faut donc définir des **critères** pour sélectionner ceux qui nous sont le plus avantageux. et qu'on va tenter d'optimiser.

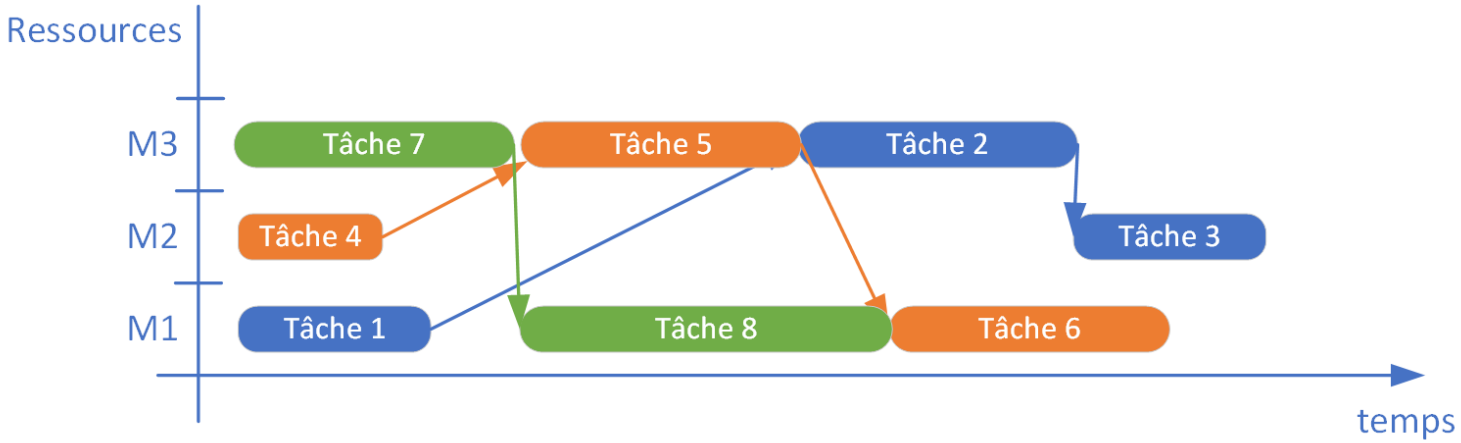
Quelques critères courants

1. La date de fin de l'ordonnancement (C_{max} appelé aussi Makespan),
2. Les différents retards (maximum, moyen, somme, nombre, etc.),
3. Les différentes avances par rapport aux dates limites fixées (maximum, moyen, somme, nombre, etc.).

Qu'est ce que l'ordonnancement ?

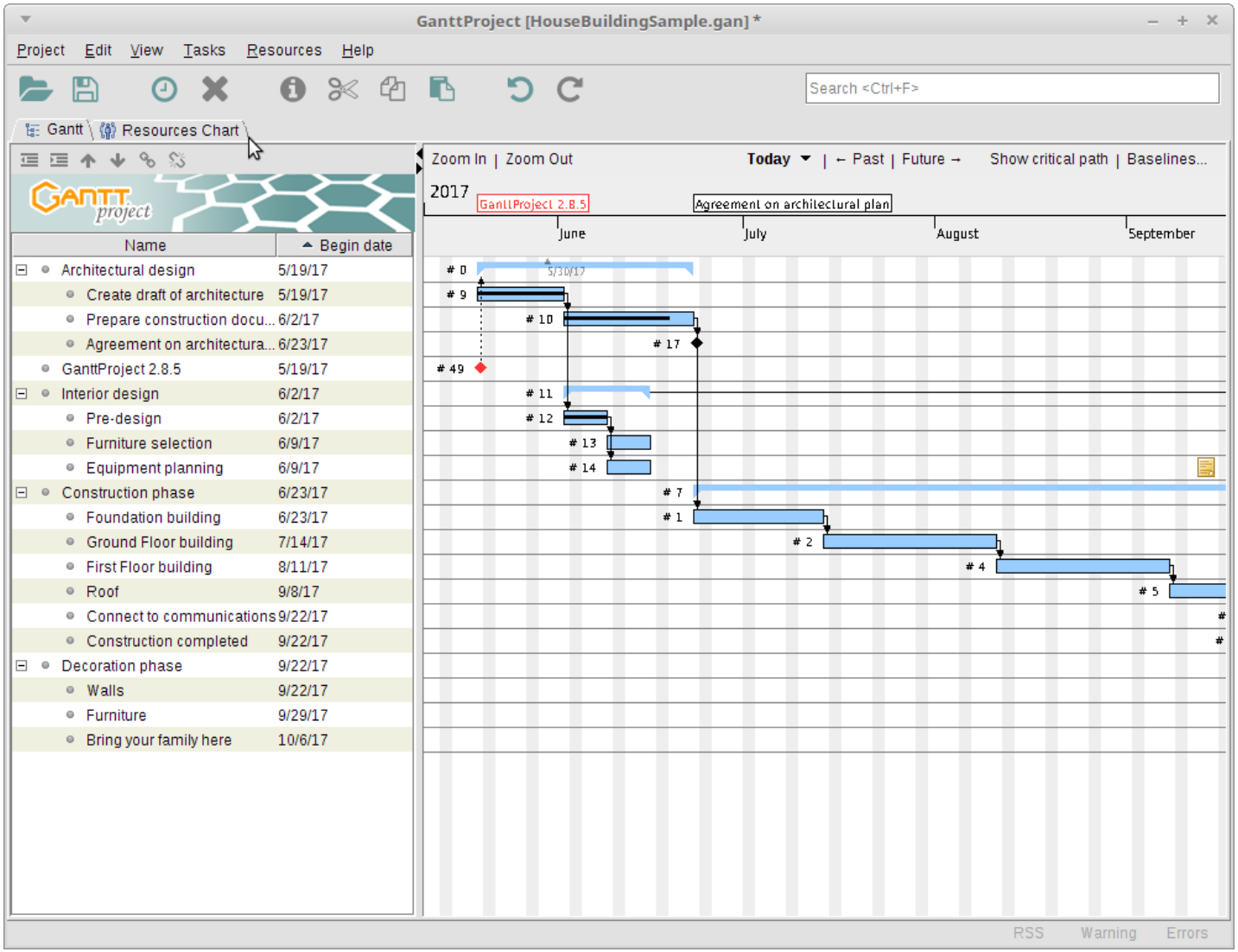
On veut minimiser le C_{max} :



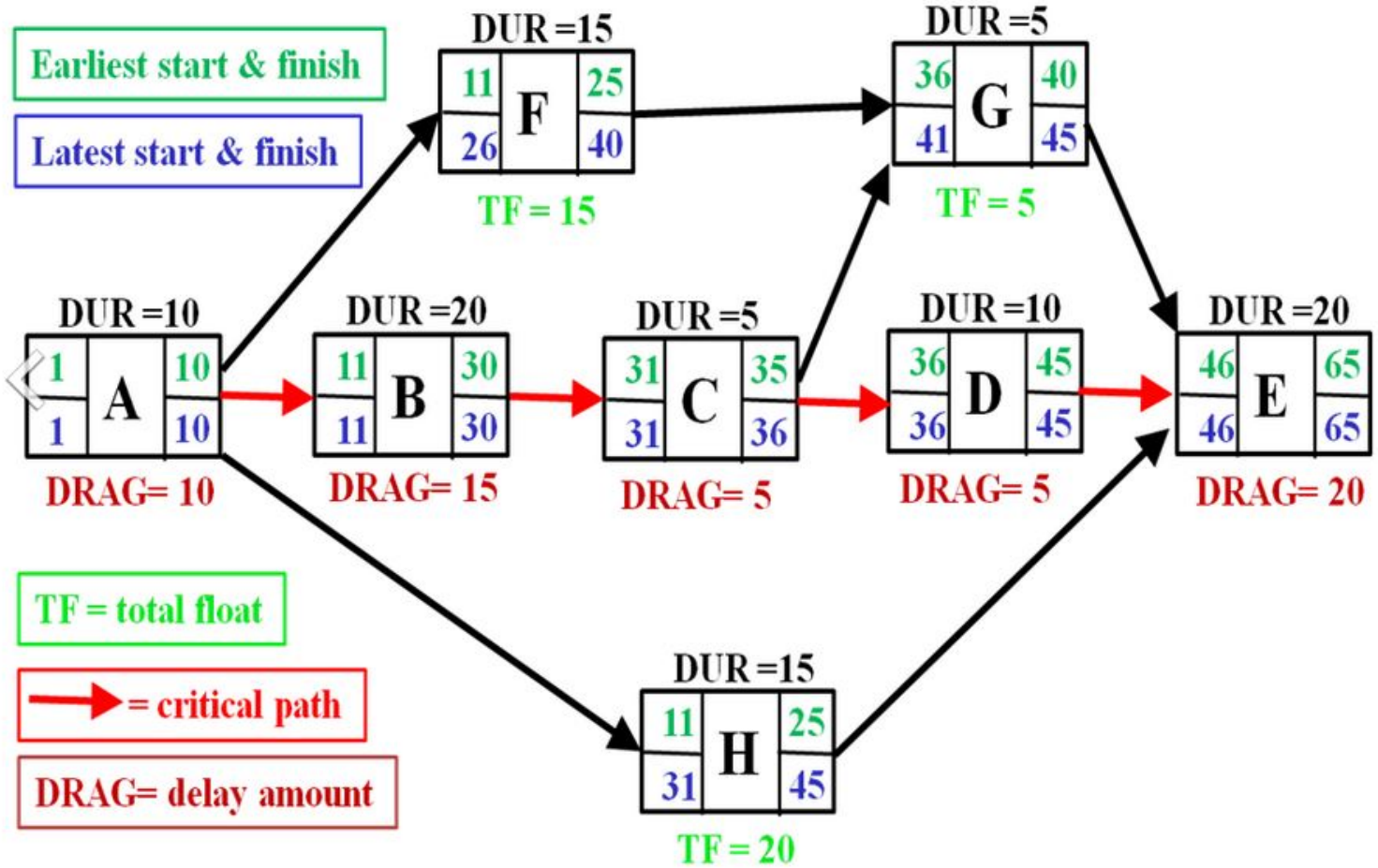


Sommaire

La gestion de projet



La gestion de projet



Problème d'atelier

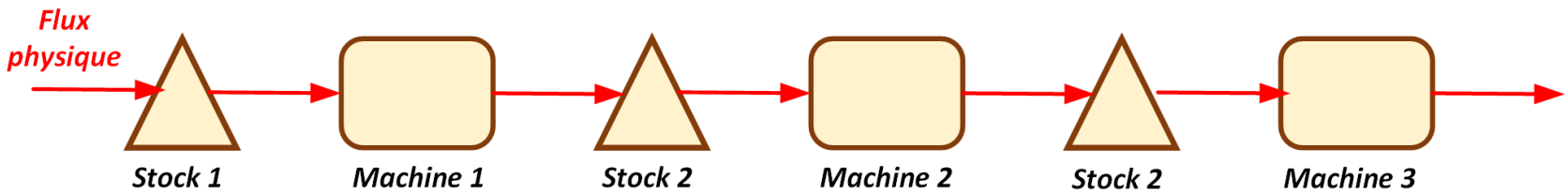
Quelques définitions

1. Un **problème d'atelier** consiste à ordonnancer la production d'un ensemble de pièces (des *jobs*) sur un ensemble de machines.
2. La production d'une pièce nécessite plusieurs opérations (*tâches*). Une opération se déroule sur une machine (ou un type de machine) et dure un certain temps.

FlowShop de permutation déterministe

Définition

1. C'est un problème d'atelier
2. Toutes les pièces passent sur toutes les machines en suivant le même chemin et en gardant l'ordre de départ (ie. passent sur les machines dans le même ordre donné)



FlowShop de permutation déterministe

Les hypothèses

1. Tous les jobs sont disponibles à la date 0
2. Les machines sont toujours disponibles
3. Les temps de traitement sont déterministes et indépendants
4. Les temps de montage et de démontage sont inclus dans les temps de traitement
5. Les temps de transport sont négligeables
6. Il n'y a pas de préemption
7. Une machine ne peut pas traiter plus d'un produit à un instant donné.

FlowShop de permutation déterministe

Les hypothèses

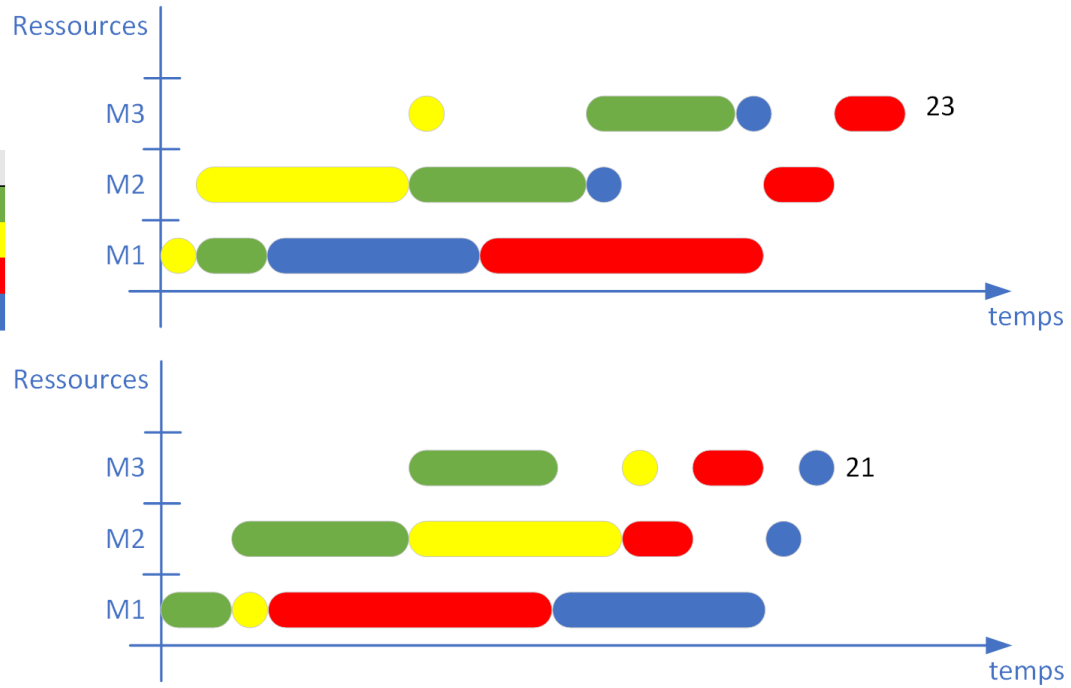
1. Un job est traité par au plus une machine à un instant donné,
2. Les produits peuvent attendre dans des stocks de capacité illimitée entre les machines,
3. Chaque job i requiert un temps de traitement t_{ij} sur la machine j,
4. Les stocks sont gérés selon la discipline FIFO

L'objectif

Minimiser le Makespan

FlowShop de permutation déterministe

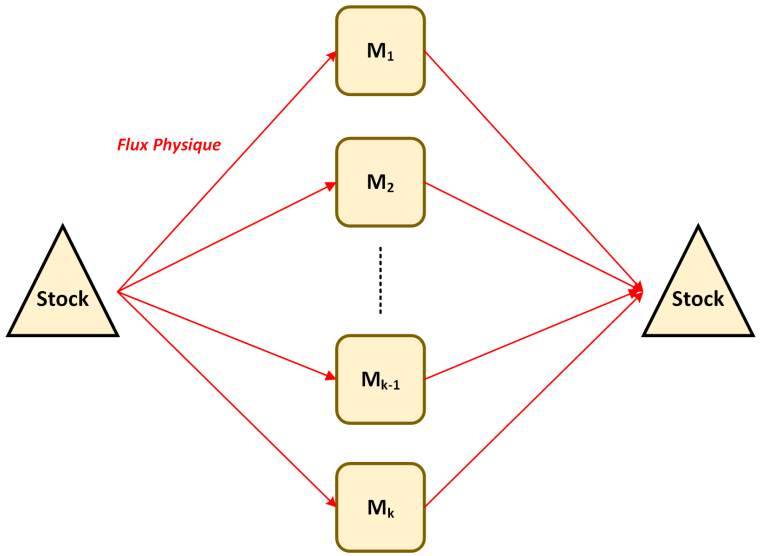
Jobs	M1	M2	M3
1	2	5	4
2	1	6	1
3	8	2	2
4	6	1	1



Machines parallèles

Définition

1. C'est un problème d'atelier
2. Les K machines sont de même type
3. Chaque pièce doit passer sur exactement une machine.



Machines parallèles

Les hypothèses

1. Tous les jobs sont disponibles à la date 0
2. Les machines sont toujours disponibles
3. Les temps de traitement sont déterministes et indépendants
4. Les temps de montage et de démontage sont inclus dans les temps de traitement
5. Les temps de transport sont négligeables
6. Il n'y a pas de préemption
7. Une machine ne peut pas traiter plus d'un produit à un instant donné.

Machines parallèles

Les hypothèses

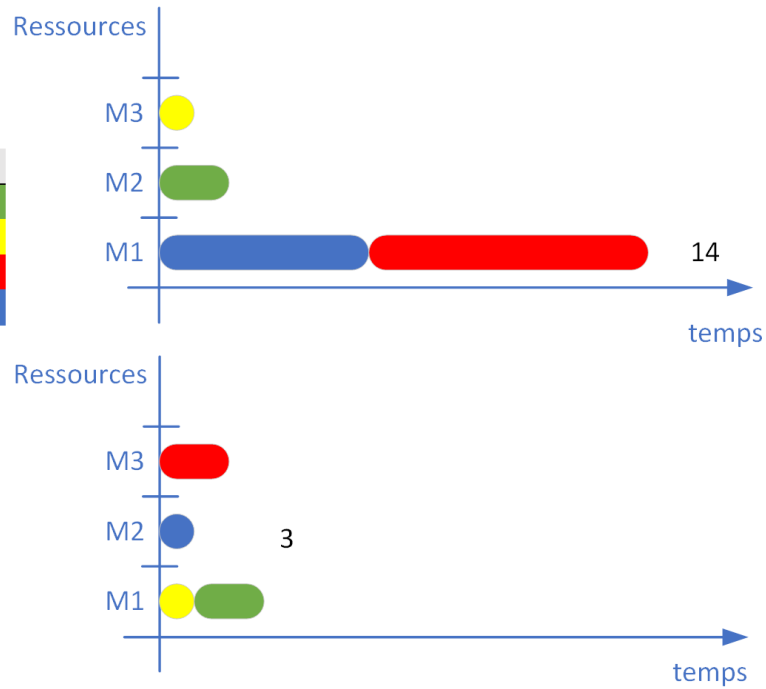
1. Un job est traité par au plus une machine à un instant donné,
2. Chaque job i requiert un temps de traitement p_{ij} sur la machine j,
3. Il n'existe pas de contraintes de précédence entre les jobs

L'objectif

Minimiser le Makespan

Machines parallèles

Jobs	M1	M2	M3
1	2	5	4
2	1	6	1
3	8	2	2
4	6	1	1



Problèmes d'atelier

Il existe beaucoup d'autres problèmes d'atelier

1. FlowShop Hybride
2. JobShop
3. OpenShop
4. ...

Sommaire

La recherche opérationnelle

La recherche opérationnelle, c'est :

La discipline des méthodes scientifiques pour aider à mieux décider (The science of better : http ://www.scienceofbetter.org)

Objectif de la Recherche Opérationnelle

- Faire de la recherche scientifique ≪ opérationnelle ≫ à l'aide des outils de l'informatique,
- Mettre au point des méthodes et les implémenter au sein d'outils logiciel pour trouver des résultats et les confronter à la réalité.

Planifier, Ordonnancer

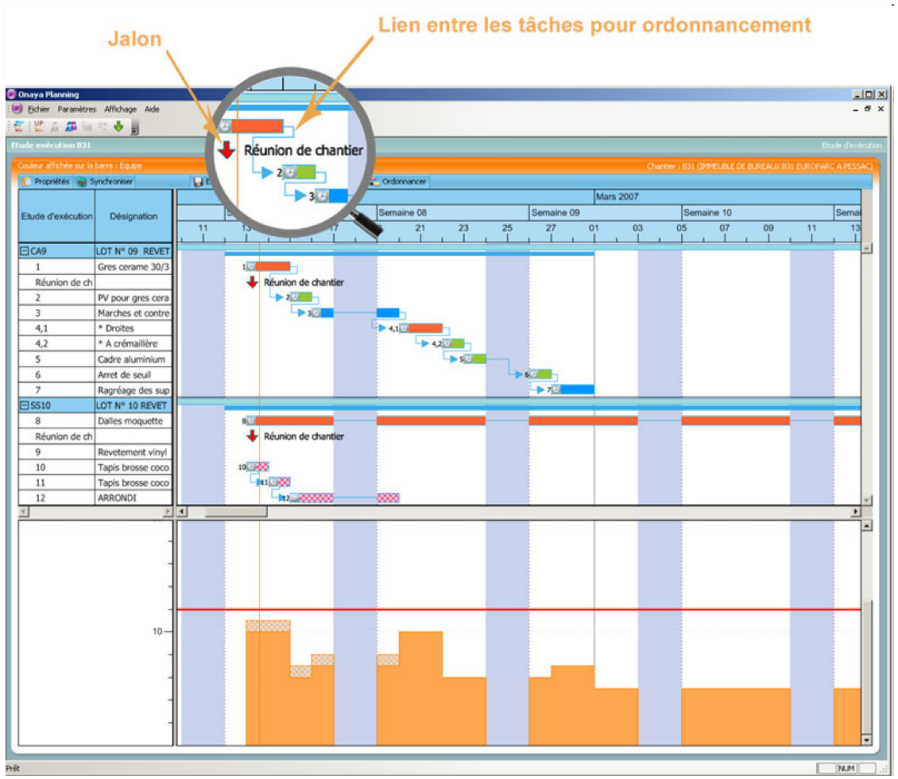


FIGURE – Planification et ordonnancement

Stocker, Gérer

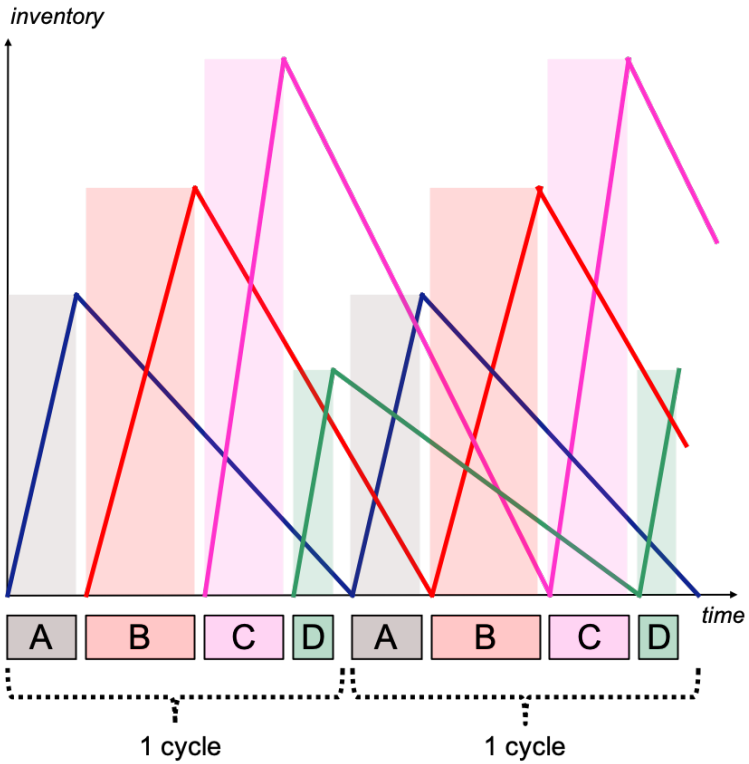
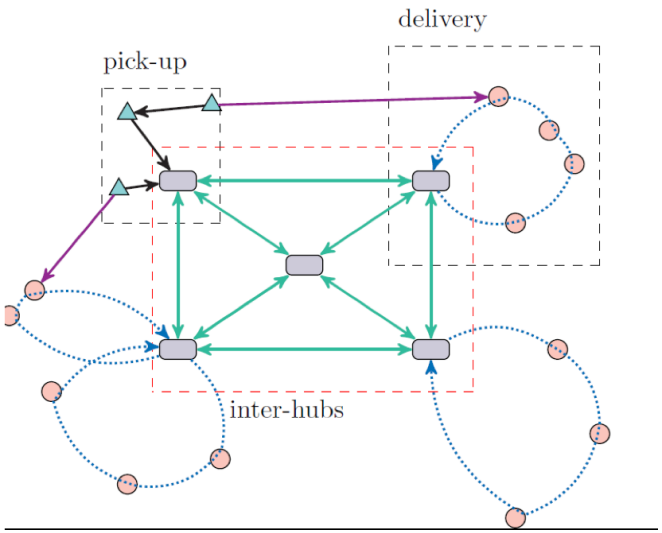


FIGURE – Gestion des stocks

Transporter



Problème du voyageur de commerce 24978 villes (2004)

FIGURE – Transporter

Emballer, Ranger

FIGURE – Ranger - Bin Packing

Mettre en relation

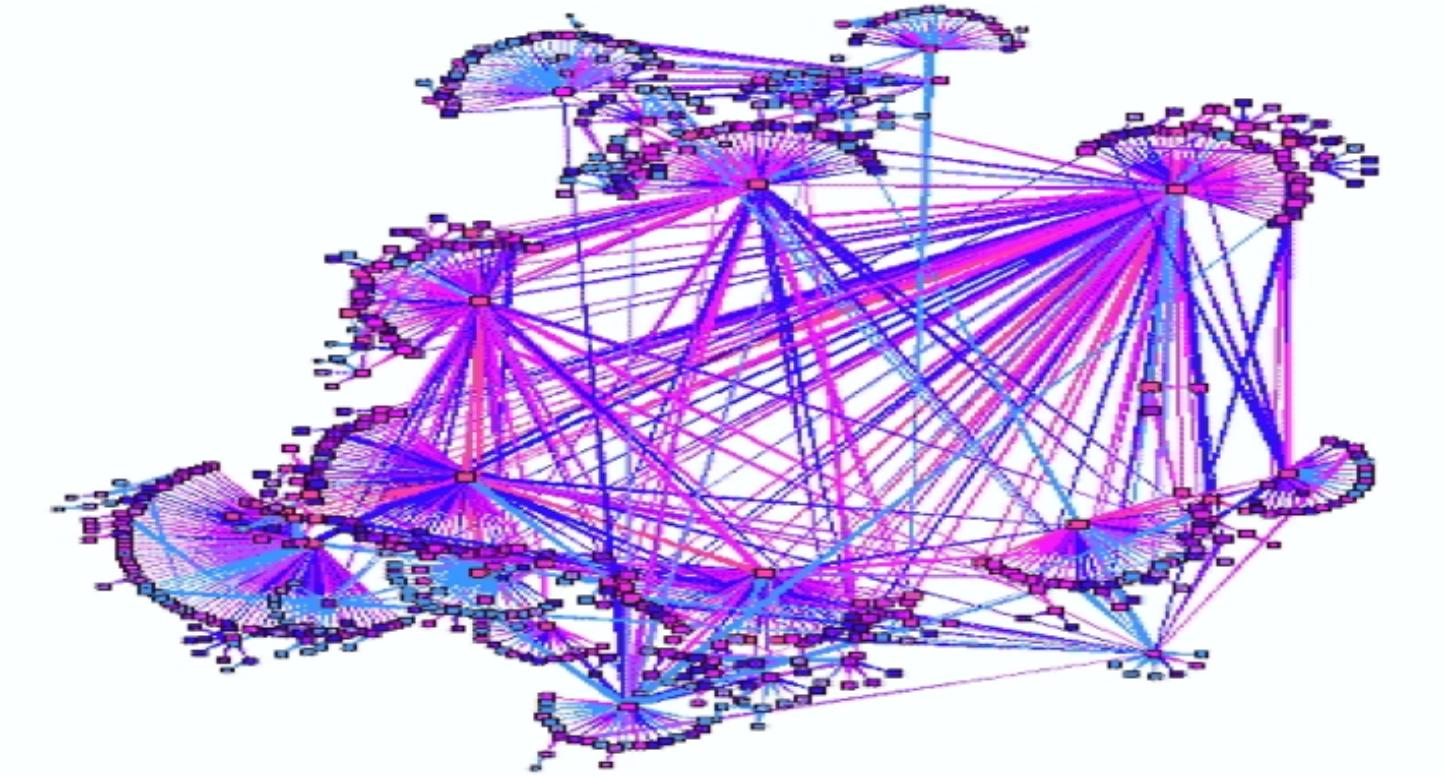
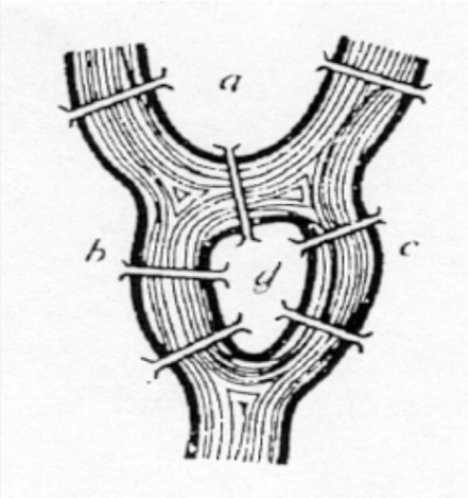


FIGURE – Mettre en relation - Réseaux mobiles

Les graphes



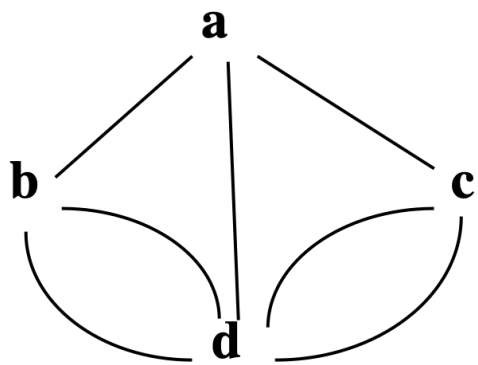


FIGURE – Les ponts de Königsberg

Existe-t'il une promenade dans les rues de Königsberg permettant, à partir d'un point de départ au choix, de passer une et une seule fois par chaque pont, et de revenir à son point de départ ?

Les graphes

Types de problème modélisés

- Circulation dans une ville
- Evolution des états d'un système
- etc.

Résolution de problèmes

- Plus court chemin
- Flot Maximal
- etc.

Les graphes

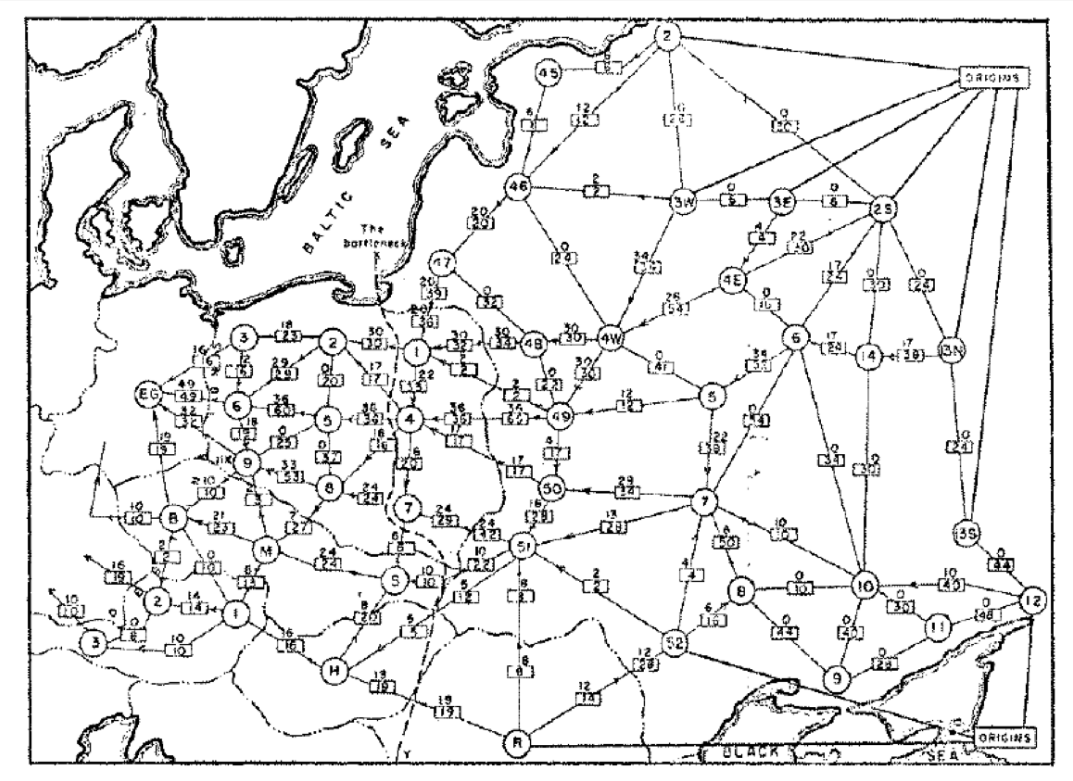


FIGURE – Problème de coupes minimales - CIA 1955

Les graphes

FIGURE – Chemin le plus rapide

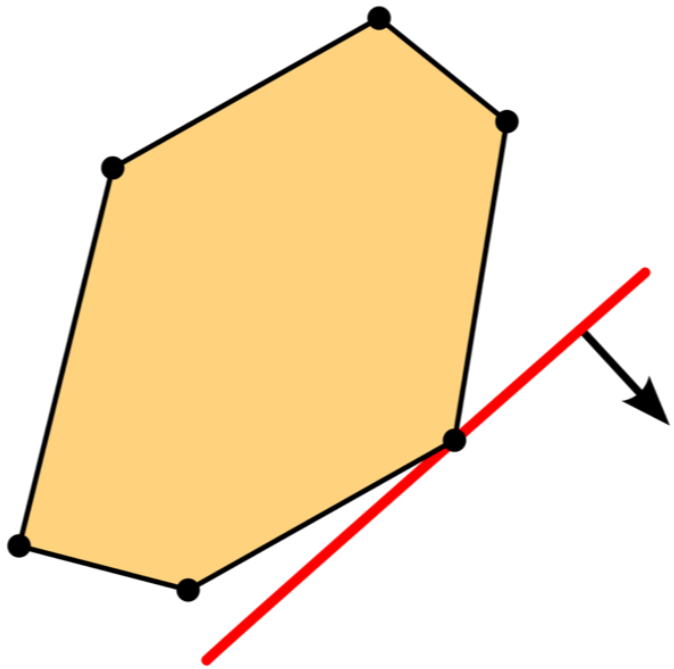
La modélisation mathématique

Programmation Linéaire

On utilise un modèle linéaire pour décrire le problème et une forme linéaire pour décrire le critère d'optimisation.
Pour cela, on définit :

- l'ensemble des paramètres du modèle (A, b, c).
- les variables de décision : x
- les contraintes du modèle : $Ax = b$ (c'est un polyèdre)
- le critère (la fonction objectif) : $\langle c, x \rangle$
- le sens d'optimisation (minimise, maximiser).

La modélisation mathématique

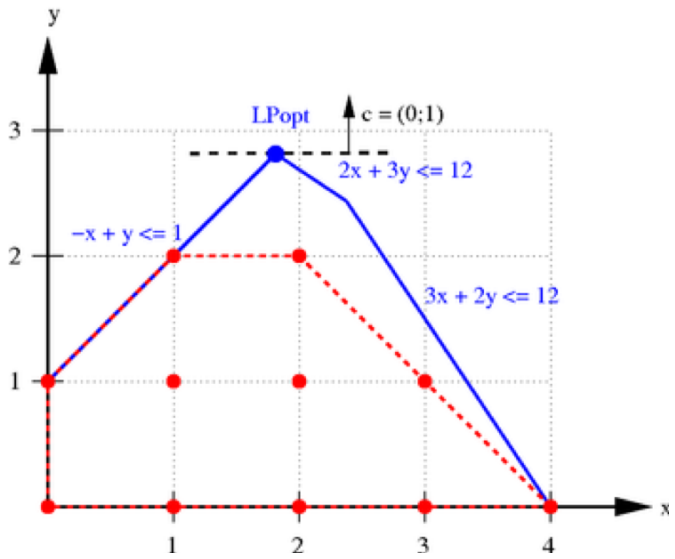


Optimiser $\langle c, x \rangle$
s.c. $Ax = b$
$x \geq 0$

FIGURE – Représentation graphique d'un programme linéaire

Un programme linéaire est facile à résoudre .
(on dit qu'il est polynomial)

La modélisation mathématique



Optimiser $\langle c, x \rangle$
s.c. $Ax = b$
$x \in \mathbb{N}^r$

FIGURE – Représentation graphique d'un programme linéaire en nombres entiers

Un programme linéaire en nombre entiers (PLNE) est souvent très difficile à résoudre .
(Dans ce cas, on dit souvent qu'il est NP Difficile)

Démarche en Recherche Opérationnelle

- On commence par essayer de déterminer si le problème est facile ou non à résoudre.
- Si le problème est facile : on exhibe un algorithme efficace (Méthode exacte)
- Si le problème est difficile :
  - Si l'instance est de « de petite taille » : on cherche une solution optimale (Méthode Exacte)
  - Si l'instance est de « de grande taille » : on cherche une solution approchée (Méthode heuristique)

Les méthodes exactes

Type de méthodes utilisées

- Simplex, Methodes par point intérieur etc. (Programmation Linéaire)
- Programmation dynamique
- Enumération implicite, Partition de l'ensemble de solution
- Méthodes ≪ arborescentes ≫(type B&B, B&C, B&P etc.)
- Algorithme adHoc.

Les méthodes approchées

Type de méthodes utilisées

- Heuristiques de bon sens (souvent basés sur de bonnes pratiques)
- Algorithmes glouton (prend la meilleur décision a un état t sans remettre en cause les états précédents)
- Recherche Locale (amélioration de solutions existantes)
- Meta-Heuristique
- ...

Sommaire

FlowShop de permutation déterministe

Les hypothèses

1. Tous les jobs sont disponibles à la date 0
2. Les machines sont toujours disponibles
3. Les temps de traitement sont déterministes et indépendants
4. Les temps de montage et de démontage sont inclus dans les temps de traitement
5. Les temps de transport sont négligeables
6. Il n'y a pas de préemption
7. Une machine ne peut pas traiter plus d'un produit à un instant donné.

FlowShop de permutation déterministe

Les hypothèses

1. Un job est traité par au plus une machine à un instant donné,
2. Les produits peuvent attendre dans des stocks de capacité illimitée entre les machines,
3. Chaque job i requiert un temps de traitement t_{ij} sur la machine j,
4. Les stocks sont gérés selon la discipline FIFO

L'objectif

Minimiser le Makespan

FlowShop de permutation déterministe

Programmation Linéaire

On utilise un modèle linéaire pour décrire le problème et une forme linéaire pour décrire le critère d'optimisation.
Pour cela, on définit :

- l'ensemble des paramètres du modèle (A, b, c).
- les variables de décision : x
- les contraintes du modèle : $Ax = b$ (c'est un polyèdre)
- le critère (la fonction objectif) : $\langle c, x \rangle$
- le sens d'optimisation (minimise, maximiser).

FlowShop de permutation déterministe

Paramètres

- N : le nombre de produits

FlowShop de permutation déterministe

Paramètres

- N : le nombre de produits
- M : le nombre de machines

FlowShop de permutation déterministe

Paramètres

- N : le nombre de produits
- M : le nombre de machines
- t_{ij} : le temps de traitement du produit $i \in \{1, \cdots, N\}$ sur la machine $j \in \{1, \cdots, M\}$

FlowShop de permutation déterministe

Variables de décision

- $z_{i,k} := 1$ si le produit i est en k^e position dans l'ordonnancement $(i,k) \in \{1, \cdots, N\}^2$,

FlowShop de permutation déterministe

Variables de décision

- $z_{i,k} := 1$ si le produit i est en k^e position dans l'ordonnancement $(i,k) \in \{1, \cdots, N\}^2$,
- $y_{k,j}$: temps d'attente du produit en position k entre la fin de son traitement sur la machine j et le début de son traitement sur la machine $j+1$, $(k,j) \in \{1, \cdots, N\} \times \{1, \cdots, M-1\}$

FlowShop de permutation déterministe

Variables de décision

- $z_{i,k} := 1$ si le produit i est en k^e position dans l'ordonnancement $(i,k) \in \{1,\cdots,N\}^2$,
- $y_{k,j}$: temps d'attente du produit en position k entre la fin de son traitement sur la machine j et le début de son traitement sur la machine $j+1$, $(k,j) \in \{1,\cdots,N\} \times \{1,\cdots,M-1\}$
- $x_{k,j}$: temps mort sur la machine j entre la fin du produit en position $k-1$ et le début du traitement du produit k, $(k,j) \in \{2,\cdots,N\} \times \{1,\cdots,M\}$

FlowShop de permutation déterministe

un job n'a qu'une seule place dans l'ordonnancement

$$\sum_{k=1}^{N} z_{i,k} = 1, \quad \forall i \in \{1, \cdots, N\}$$

à une place dans l'ordonnancement ne correspond qu'un job

$$\sum_{i=1}^{N} z_{i,k} = 1, \quad \forall k \in \{1, \cdots, N\}$$

FlowShop de permutation déterministe

Contrainte difficile à modéliser

$$x_{k+1,j} + \sum_{i=1}^{N} z_{i,k+1} t_{i,j} + y_{k+1,j} = \sum_{i=1}^{N} z_{i,k} t_{i,j+1} + x_{k+1,j+1} + y_{k,j}$$
$$\forall (j,k) \in \{1, \cdots, N-1\} \times \{1, \cdots, M-1\}$$

- Les produits sont tous traités dans le même ordre

FlowShop de permutation déterministe

Contrainte difficile à modéliser

$$x_{k+1,j} + \sum_{i=1}^{N} z_{i,k+1} t_{i,j} + y_{k+1,j} = \sum_{i=1}^{N} z_{i,k} t_{i,j+1} + x_{k+1,j+1} + y_{k,j}$$
$$\forall (j,k) \in \{1, \cdots, N-1\} \times \{1, \cdots, M-1\}$$

- Les produits sont tous traités dans le même ordre
- une machine ne peut traiter qu'un produit à la fois

FlowShop de permutation déterministe

Contrainte difficile à modéliser

$$x_{k+1,j} + \sum_{i=1}^{N} z_{i,k+1} t_{i,j} + y_{k+1,j} = \sum_{i=1}^{N} z_{i,k} t_{i,j+1} + x_{k+1,j+1} + y_{k,j}$$
$$\forall (j,k) \in \{1, \cdots, N-1\} \times \{1, \cdots, M-1\}$$

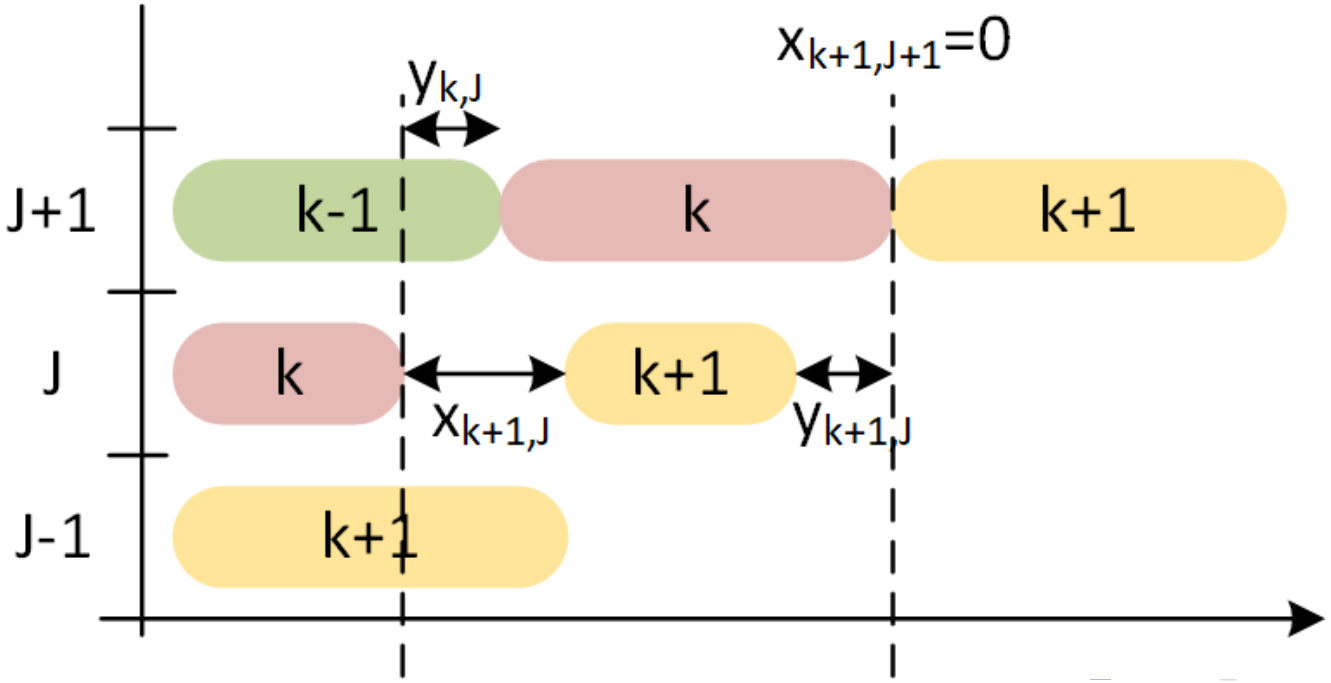
- Les produits sont tous traités dans le même ordre
- une machine ne peut traiter qu'un produit à la fois
- à un instant donné, un produit est traité au maximum pour une seule machine.

FlowShop de permutation déterministe

Contrainte difficile à modéliser

$$x_{k+1,j} + \sum_{i=1}^{N} z_{i,k+1} t_{i,j} + y_{k+1,j} = \sum_{i=1}^{N} z_{i,k} t_{i,j+1} + x_{k+1,j+1} + y_{k,j}$$

$$\forall (j,k) \in \{1, \cdots, N-1\} \times \{1, \cdots, M-1\}$$

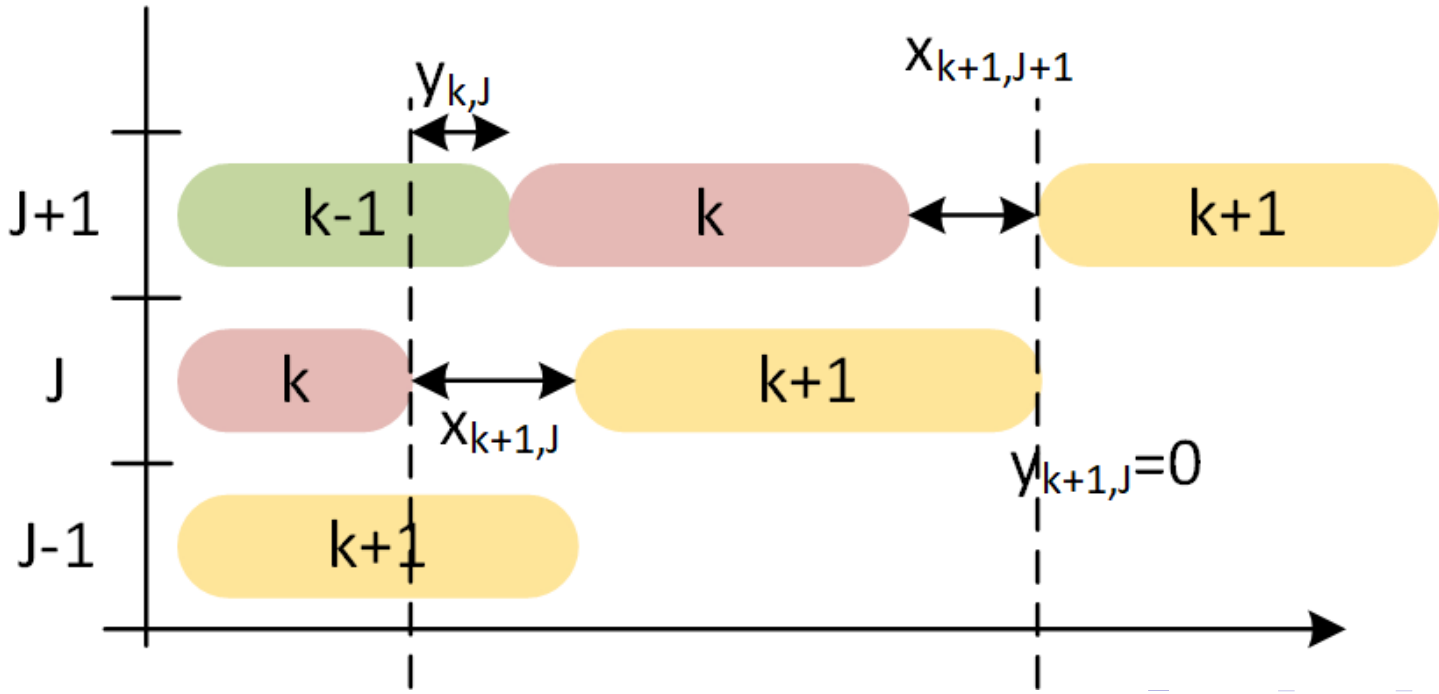


FlowShop de permutation déterministe

Contrainte difficile à modéliser

$$x_{k+1,j} + \sum_{i=1}^{N} z_{i,k+1} t_{i,j} + y_{k+1,j} = \sum_{i=1}^{N} z_{i,k} t_{i,j+1} + x_{k+1,j+1} + y_{k,j}$$

$$\forall (j,k) \in \{1, \cdots, N-1\} \times \{1, \cdots, M-1\}$$



FlowShop de permutation déterministe

Contrainte difficile à modéliser

$$x_{1,j} + \sum_{i=1}^{N} z_{i,1} t_{i,j} + y_{1,j} = x_{1,j+1} \ \forall j \in \{1, \cdots, M-1\}$$

même contrainte que précédemment mais pour le premier produit ordonnancé.

FlowShop de permutation déterministe

Positivité et intégrité

$$x_{k,j} \geq 0 \quad \forall (j,k) \in \{1,\cdots,N\} \times \{1,\cdots,M\}$$
$$y_{k,j} \geq 0 \quad \forall (j,k) \in \{1,\cdots,N\} \times \{1,\cdots,M\}$$
$$z_{i,j} \in \{0,1\} \quad \forall (i,j) \in \{1,\cdots,N\}^2$$

minimiser le makespan = minimiser les temps morts sur la dernière machine

$$\min \sum_{k=1}^{N} x_{k,M}$$

FlowShop de permutation déterministe

Traduction OPL

Paramètres

Les paramètres du modèle

Variables de décisions

Les variables de décisions

Le modèle mathématique

Minimize obj
subject to
{
    Contraintes
}

FlowShop de permutation déterministe

Paramètres

- N : le nombre de produits
- M : le nombre de machines
- t_{ij} : le temps de traitement du produit $i \in \{1, \cdots, N\}$ sur la machine $j \in \{1, \cdots, M\}$

Traduction OPL

```
int N=...; //Nombre de produits
int M=...; //Nombre de machines
range I=1..N; //Ensemble de produits
range K=1..N; // Ensemble de positions dans l'ordonnancement
range J=1..M; //Ensemble de machines
float t[I][J]=...; // t[i][j] temps de traitement du produit i sur la machine j
```

FlowShop de permutation déterministe

Variables de décision

- $z_{i,k}$: $= 1$ si le produit i est en k^e position dans l'ordonnancement $(i,k) \in \{1,\cdots,N\}^2$,
- $y_{k,j}$: temps d'attente du produit en position k entre la fin de son traitement sur la machine j et le début de son traitement sur la machine $j+1$, $(k,j) \in \{1,\cdots,N\} \times \{1,\cdots,M-1\}$
- $x_{k,j}$: temps mort sur la machine j entre la fin du produit en position $k-1$ et le début du traitement du produit k, $(k,j) \in \{2,\cdots,N\} \times \{1,\cdots,M\}$

Traduction OPL

```
dvar boolean z[I][K] ; // z[i][k]= 1 si le produit i est à la position k
dvar float+ y[K][J] ; // temps d'attente ...
dvar float+ x[K][J] ; // temps mort ...
```

FlowShop de permutation déterministe

un job n'a qu'une seule place dans l'ordonnancement

$$\sum_{k=1}^{N} z_{i,k} = 1, \quad \forall i \in \{1, \cdots, N\}$$

à une place dans l'ordonnancement ne correspond qu'un job

$$\sum_{i=1}^{N} z_{i,k} = 1, \quad \forall k \in \{1, \cdots, N\}$$

Traduction OPL

```
forall(i in I)
      sum(k in K) z[i][k]==1;
forall(k in K)
      sum(i in I) z[i][k]==1;
```

FlowShop de permutation déterministe

Contrainte difficile à modéliser

$$x_{k+1,j} + \sum_{i=1}^{N} z_{i,k+1} t_{i,j} + y_{k+1,j} = \sum_{i=1}^{N} z_{i,k} t_{i,j+1} + x_{k+1,j+1} + y_{k,j}$$
$$\forall (j,k) \in \{1, \cdots, N-1\} \times \{1, \cdots, M-1\}$$

Traduction OPL

```
forall(j in 1..N-1, k in 1..M-1)
      x[k+1][j]+sum(i in I)(z[i][k+1]*t[i][j])+y[k+1][j]
      -(sum(i in I)(z[i][k]*t[i][j+1])-x[k+1][j+1]-y[k][j]==0 ;
```

FlowShop de permutation déterministe

Contrainte difficile à modéliser

$$x_{1,j} + \sum_{i=1}^{N} z_{i,1} t_{i,j} + y_{1,j} = x_{1,j+1} \ \forall j \in \{1, \cdots, M-1\}$$

Traduction OPL

```
forall(j in 1..N-1)
    x[1][j]+sum(i in I)(z[i][1]*t[i][j])+y[1][j]-x[1][j+1]==0;
```

FlowShop de permutation déterministe

Positivité et intégrité

$x_{k,j} \geq 0 \quad \forall (j,k) \in \{1, \cdots, N\} \times \{1, \cdots, M\}$
$y_{k,j} \geq 0 \quad \forall (j,k) \in \{1, \cdots, N\} \times \{1, \cdots, M\}$
$z_{i,j} \in \{0,1\} \quad \forall (i,j) \in \{1, \cdots, N\}^2$

minimiser le makespan = minimiser les temps morts sur la dernière machine

$$\min \sum_{k=1}^{N} x_{k,M}$$

Traduction OPL

```
minimize sum(k in K) x[k][M] ;
```

FlowShop de permutation déterministe

```
/*********************************************
 * OPL 12.8.0.0 Model
 * Author: Ecole MACS Bermudes-SED
 * Creation Date: 27 mars 2019 at 14:30:25
 *********************************************/
/*********************************************
 * Paramètres
 *********************************************/
int N=...; //Nombre de produits
int M=...; //Nombre de machines
range I=1..N; //Ensemble de jobs
range K=1..N; // Ensemble de positions dans l'ordonnancement
range J=1..M; //Ensemble de machines
float t[I][J]=...; // t[i][j] temps de traitement du produit i sur la machine j
/*********************************************
 * Variables de décision
 *********************************************/
dvar boolean z[I][K]; // z[j][k] 1 si le produit i est à la position k
dvar float+  y[K][J]; // temps d'attente ...
dvar float+  x[K][J]; // temps mort ...
/*********************************************
 * Modèle mathématique
 *********************************************/
minimize sum(k in K) x[k][M];
subject to
{
        forall(i in I)
            sum(k in K) z[i][k]==1;
        forall(k in K)
            sum(i in I) z[i][k]==1;
        forall(j in 1..N-1, k in 1..M-1)
            x[k+1][j]+sum(i in I)(z[i][k+1]*t[i][j])+y[k+1][j]-sum(i in I)(z[i][k]*t[i][j+1])-x[k+1][j+1]-y[k][j]==0;
        forall(j in 1..N-1)
            x[1][j]+sum(i in I)(z[i][1]*t[i][j])+y[1][j]-x[1][j+1]==0;
}
```

FIGURE – Modélisation sous OPL

Machines parallèles

Les hypothèses

1. Tous les jobs sont disponibles à la date 0
2. Les machines sont toujours disponibles
3. Les temps de traitement sont déterministes et indépendants
4. Les temps de montage et de démontage sont inclus dans les temps de traitement
5. Les temps de transport sont négligeables
6. Il n'y a pas de préemption
7. Une machine ne peut pas traiter plus d'un produit à un instant donné.

Machines parallèles

Les hypothèses

1. Un job est traité par au plus une machine à un instant donné,
2. Chaque job i requiert un temps de traitement p_{ij} sur la machine j,
3. Il n'existe pas de contraintes de précédence entre les jobs

L'objectif

Minimiser le Makespan

Machines parallèles

A vous de jouer

Machines parallèles

Paramètres

- N : le nombre de produits

Variables de décision

Machines parallèles

Paramètres

- N : le nombre de produits
- K : le nombre de machines

Variables de décision

Machines parallèles

Paramètres

- N : le nombre de produits
- K : le nombre de machines
- p_{jk} : temps de traitement du job j sur la machine k, $(j, k) \in \{1, \ldots, N\} \times \{1, \ldots, K\}$

Variables de décision

Machines parallèles

Paramètres

- N : le nombre de produits
- K : le nombre de machines
- p_{jk} : temps de traitement du job j sur la machine k, $(j,k) \in \{1,\ldots,N\} \times \{1,\ldots,K\}$

Variables de décision

- $x_{jk} = \begin{cases} 1 & \text{si } j \text{ est exécuté sur la machine } k \\ 0 & \text{sinon} \end{cases}$

Machines parallèles

Paramètres

- N : le nombre de produits
- K : le nombre de machines
- p_{jk} : temps de traitement du job j sur la machine k, $(j,k) \in \{1,\ldots,N\} \times \{1,\ldots,K\}$

Variables de décision

- $x_{jk} = \begin{cases} 1 & \text{si } j \text{ est exécuté sur la machine } k \\ 0 & \text{sinon} \end{cases}$
- C_{max} : valeur du Makespan

Machines parallèles

Contraintes

- $\sum_{k=1}^{K} x_{jk} = 1 \quad \forall j \in \{1, \ldots, N\}$

Objectif

min C_{max}

Machines parallèles

Contraintes

- $\sum_{k=1}^{K} x_{jk} = 1 \quad \forall j \in \{1, \ldots, N\}$
- $C_{max} - \sum_{j=1}^{N} p_{jk} x_{jk} \geq 0 \quad \forall k \in \{1, \ldots, K\}$

Objectif

min C_{max}

Machines parallèles

Contraintes

- $\sum_{k=1}^{K} x_{jk} = 1 \quad \forall j \in \{1, \ldots, N\}$
- $C_{max} - \sum_{j=1}^{N} p_{jk} x_{jk} \geq 0 \quad \forall k \in \{1, \ldots, K\}$
- $C_{max} \geq 0, x_{jk} \in \{0, 1\} \quad \forall (j, k) \in \{1, \ldots, N\} \times \{1, \ldots, K\}$

Objectif

min C_{max}

Sommaire

Robustesse

Soit le programme linéaire en variables mixtes suivant :

$$\mathcal{P} \left| \begin{array}{ll} \max\limits_{x} & \langle c, x \rangle \\ & \sum\limits_{j=1}^{n} a_{ij} x_j \leq b_i, i = 1, \cdots, m. \\ & x_j \in \mathbb{N}, j = 1, \cdots, p, \\ & x_j \in \mathbb{R}_+, j = p+1, \cdots, n. \end{array} \right.$$

On suppose que les incertitudes portent uniquement sur les coefficients a_{ij} . Plus précisemment, nous supposons que chaque coefficient a_{ij} prend ses valeurs dans un **intervalle borné** de la forme $[\bar{a}_{ij} - \hat{a}_{ij}, \bar{a}_{ij} + \hat{a}_{ij}]$, autrement dit, il existe une variable aléatoire ζ_{ij} à valeur dans $[-1, 1]$ tel que

$$a_{ij} = \bar{a}_{ij} + \zeta_{ij} \hat{a}_{ij}.$$

Robustesse

On peut donc reformuler le programme linéaire de la façon suivante :

$$\mathcal{P} \left| \begin{array}{ll} \max\limits_{x} & \langle c, x \rangle \\ & \sum\limits_{j=1}^{n} \bar{a}_{ij} x_j + \sum\limits_{j=1}^{n} \zeta_{ij} \hat{a}_{ij} x_j \leq b_i, i = 1, \cdots, m. \\ & x_j \in \mathbb{N}, j = 1, \cdots, p, \\ & x_j \in \mathbb{R}_+, j = p+1, \cdots, n. \end{array} \right.$$

Robustesse

$$\mathcal{P}' \left| \begin{array}{ll} \max\limits_{x} & \langle c, x \rangle \\ & \sum\limits_{j=1}^{n} \bar{a}_{ij} x_j + \beta_i(x) \leq b_i, i = 1, \cdots, m. \\ & x_j \in \mathbb{N}, j = 1, \cdots, p, \\ & x_j \in \mathbb{R}_+, j = p+1, \cdots, n. \end{array} \right.$$

Approches en ligne

$$\beta_i(x) = \max_{\|\zeta_i\|_l \leq \Omega_i} \left(\sum_{j=1}^{n} \hat{a}_{ij} \zeta_{ij} x_j \right), i = 1, \cdots, m$$

où $l = 1, 2$ selon la norme utilisée.

Robustesse

Bertsimas et Sim

On utilise la norme $\|\|\|_1$: avantage, on reste linéaire (avec $\|\|\|_2$ on devient quadratique)

$$\beta_i(x) = \max_{\|\zeta_i\|_1 \leq \Omega_i} \left(\sum_{j=1}^{n} \hat{a}_{ij} \zeta_{ij} x_j \right), i = 1, \cdots, m$$

$$\begin{aligned} \beta_i(x^*) = \quad \max \quad & \sum_{j=1}^{n} \left(\hat{a}_{ij} x_j^* \right) \zeta_{ij} \\ & \sum_{j=1}^{n} \zeta_{ij} \leq \Omega_i. \\ & \zeta_{ij} \leq 1, j = 1, \cdots, n. \\ & \zeta_{ij} \geq 0, j = 1, \cdots, n. \end{aligned}$$

Robustesse

Bertsimas et Sim

C'est équivalent au Programme Linéaire suivant (par dualité)

$$\begin{aligned} \beta_i(x^*) = \quad \min \quad & \sum_{j=1}^{n} p_{ij} + z_i\Omega_i \\ & z_i + p_{ij} \geq \hat{a}_{ij}x_j^*, j = 1, \cdots, n. \\ & p_{ij} \geq 0, j = 1, \cdots, n. \\ & z_i \geq 0. \end{aligned}$$

Robustesse

Bertsimas et Sim

$$\mathcal{P}_1' \left| \begin{array}{ll} \max\limits_{x} & \langle c, x \rangle \\ & \sum\limits_{j=1}^{n} \bar{a}_{ij} x_j + \sum\limits_{j=1}^{n} p_{ij} + z_i \Omega_i \leq b_i, i = 1, \cdots, m. \\ & z_i + p_{ij} \geq \hat{a}_{ij} x_j^*, i = 1, \cdots, m, j = 1, \cdots, n. \\ & p_{ij} \geq 0, i = 1, \cdots, m, j = 1, \cdots, n. \\ & z_i \geq 0. \\ & x_j \in \mathbb{N}, j = 1, \cdots, p, \\ & x_j \in \mathbb{R}_+, j = p+1, \cdots, n. \end{array} \right.$$

A vous de jouer : robustifier le modèle à machines parallèles

Robustesse

Bertsimas et Sim

$$
\begin{aligned}
\min \quad & C_{max} \\
& \sum_{k=1}^{K} x_{jk} = 1, j = 1, \cdots, N. \\
& C_{max} - \sum_{j=1}^{N} \bar{p}_{jk} x_{jk} - \Omega_k z_k - \sum_{j=1}^{N} a_{jk} \geq 0, k \in \{1, \ldots, K\}. \\
& z_k + a_{jk} \geq \hat{p}_{jk} x_{jk}, (j,k) \in \{1, \ldots, N\} \times \{1, \ldots, K\}. \\
& a_{jk} \geq 0, (j,k) \in \{1, \ldots, N\} \times \{1, \ldots, K\}. \\
& z_k \geq 0, k \in \{1, \ldots, K\}, \\
& C_{max} \geq 0, x_{jk} \in \{0,1\}, (j,k) \in \{1, \ldots, N\} \times \{1, \ldots, K\}.
\end{aligned}
$$

Robustesse

Bertsimas et Sim

Problème : comment calibrer les Ω_i de façon à ajuster le niveau de robustesse ? Une réponse : Les Systèmes à Événements Discrets