Optimisation dans les graphes : modèles, algorithmes et applications

François Clautiaux, Boris Detienne

Réseau Mexico, 12-13 novembre 2018

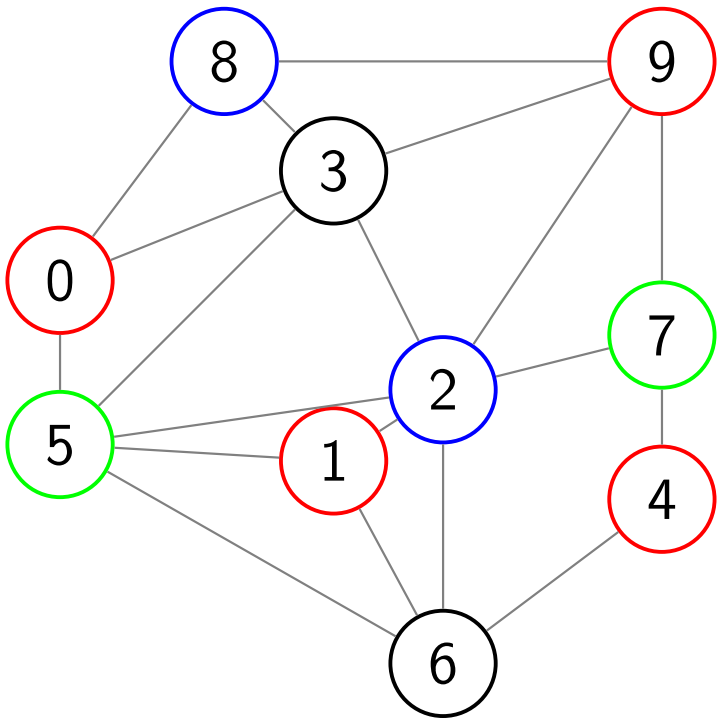
Problématiques traités dans cette présentation

Quels sont les problèmes classiques d'optimisation dans les graphes ?

Comment résoudre des problèmes difficiles ?

Quelles techniques pour traiter les modèles de très grande taille ?

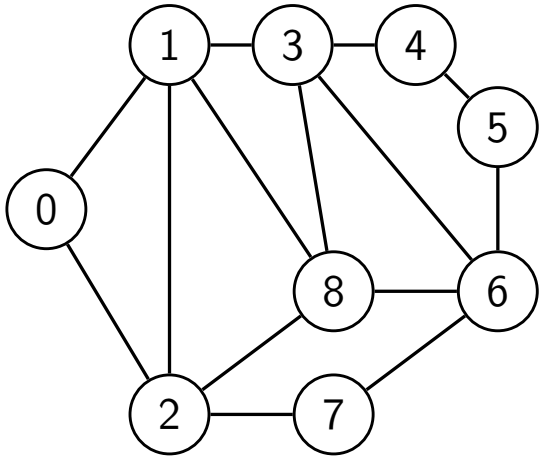
Comment intègre-t-on l'incertitude dans les problèmes de graphe ?



Notion de graphe

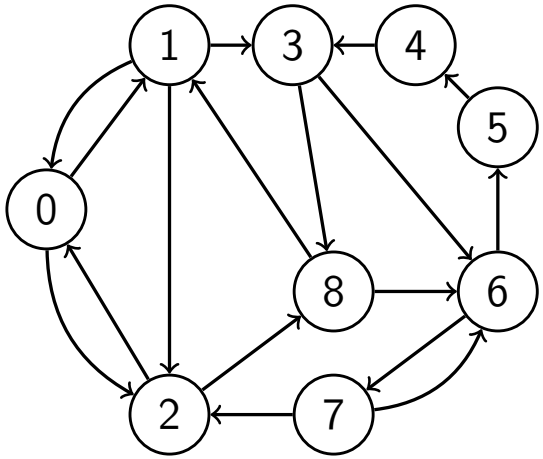
Definition

Un **graphe non orienté** est une paire (V, A) où V est un ensemble de n *sommets*, et $A \subseteq V \times V$ est un ensemble de m paires de sommets appelées *arêtes*.



Definition

Un **graphe orienté** est une paire (V, A) où V est un ensemble de n *sommets*, et $A \subseteq V \times V$ est un ensemble de m couples de sommets appelées *arcs*.



Les arcs/arêtes définissent une relation entre les sommets.

Notions élémentaires

Un **chemin** est une suite d'arcs $(x_1, x_2), (x_2, x_3), \ldots, (x_{k-1}, x_k)$. On parle de **chaîne** dans les graphes non orientés.
Si $x_1 = x_k$ alors on parle de **circuit** (resp. **cycle**).

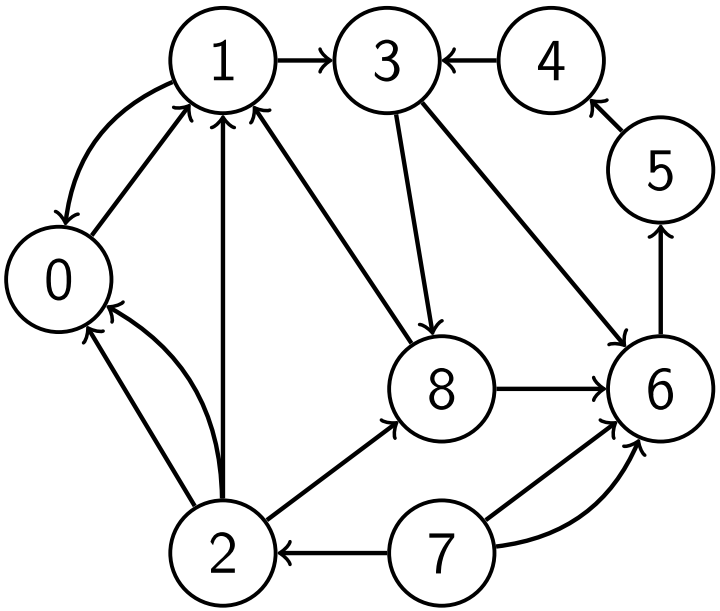
Soit $G = (V, E)$ un graphe. $W \subseteq V$ est une **composante connexe** (resp. **fortement connexe**) s'il existe une chaîne (resp. un chemin) entre toute paire de sommet de $W \times W$.

Pour $W \subseteq V$, le graphe $(W, E \cap W^2)$ est appelé **sous-graphe** de W.

Si $F \subseteq E$, le graphe (V, F) est appelé **graphe partiel** de G.

$G = (V, V^2)$ est appelé **graphe complet**. Un sous-graphe complet est appelé **clique**.

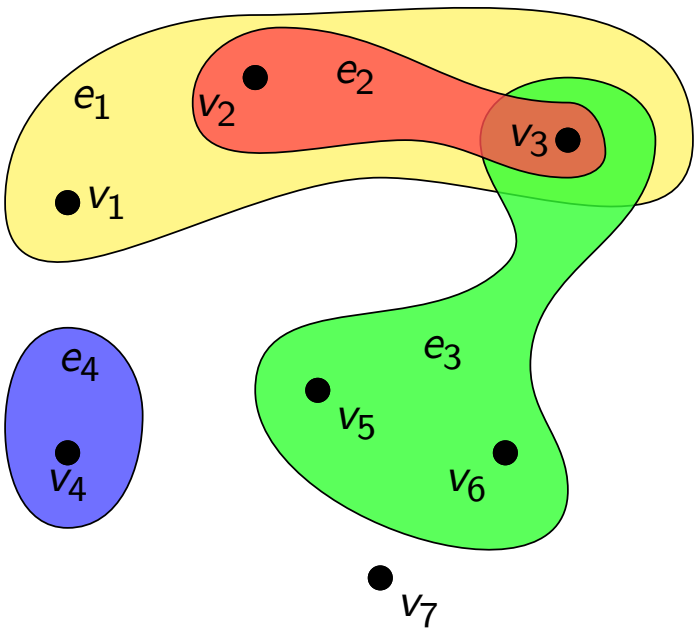
Extensions des graphes : multi-graphes



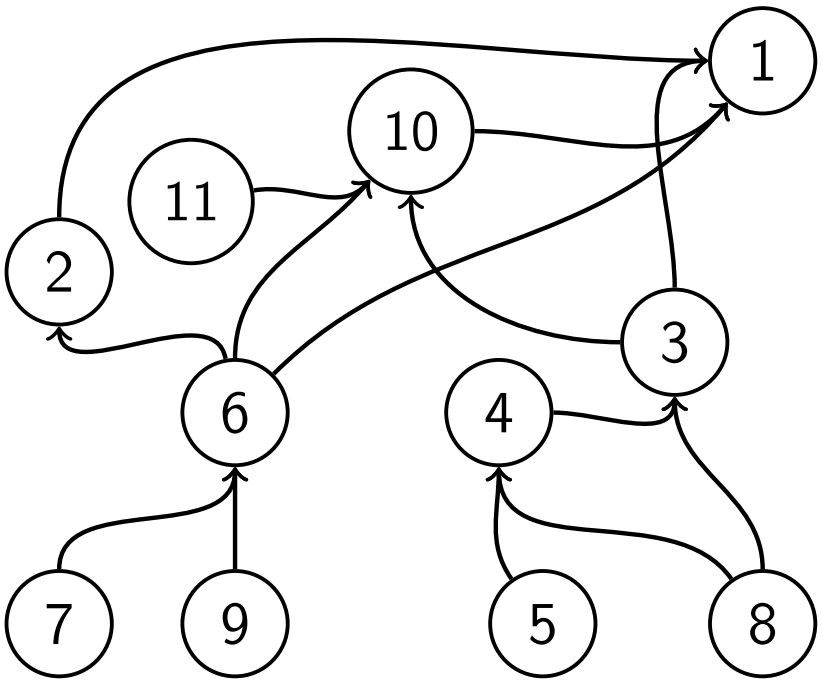
Multi-graphes

Dans un multi-graphe, il peut exister plusieurs arcs/arêtes entre deux sommets.

Extensions des graphes : hypergraphes



Hyper-graphes non orientés



Hypergraphes orientés

Graphes et optimisation
Modélisation et domaines d'application

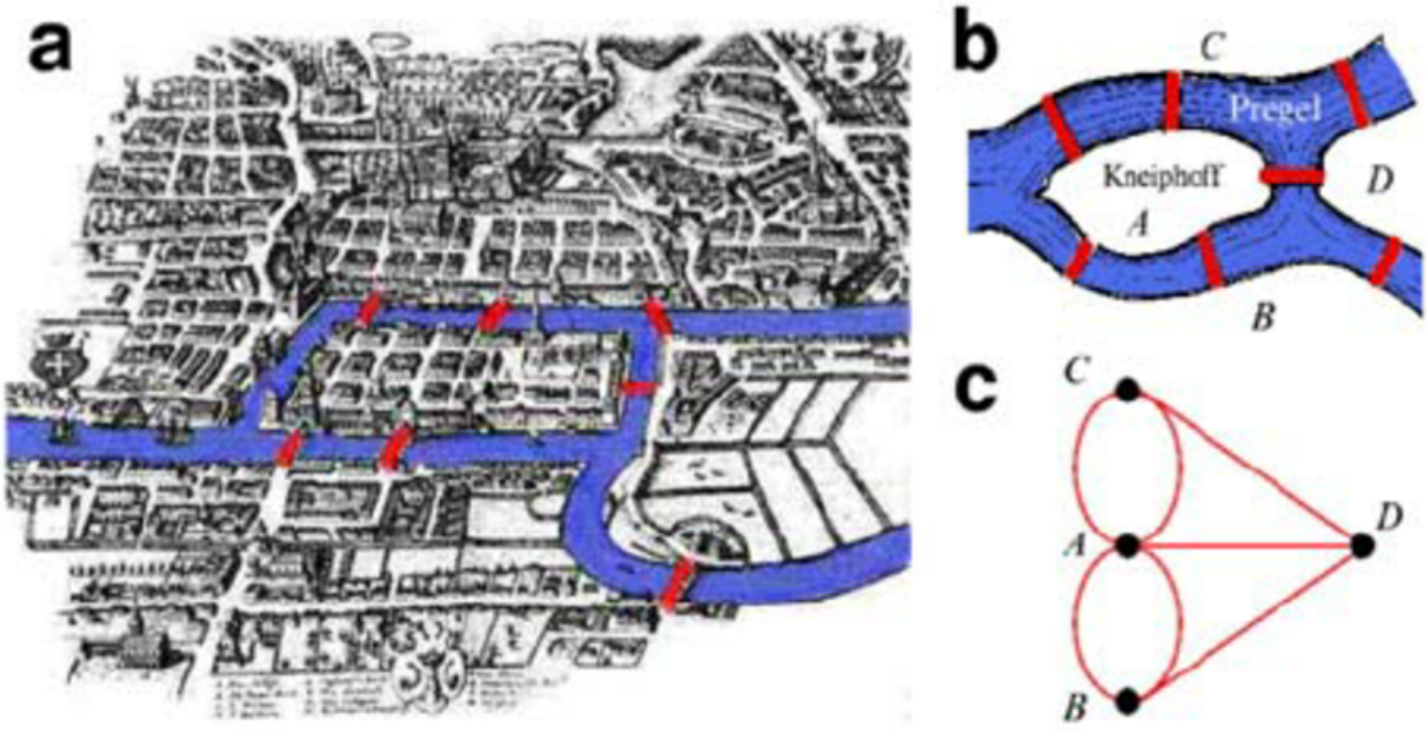
Problèmes classiques d'optimisation dans les graphes

Cas d'étude : le problème de voyageur de commerce

Gestion de l'incertitude dans les problèmes de graphe

Un peu d'histoire (1)

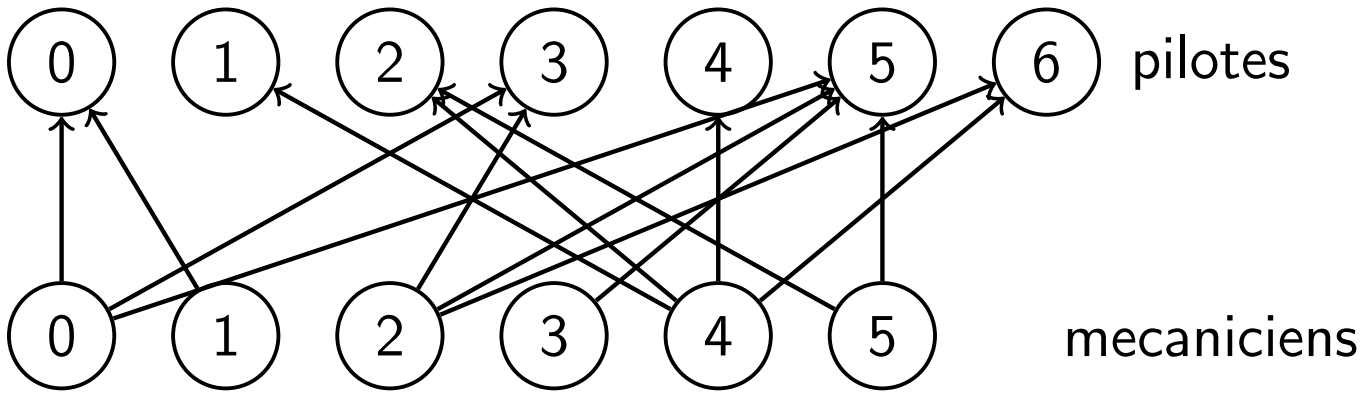
Ponts de Konigsberg



Source : Nunes Maral

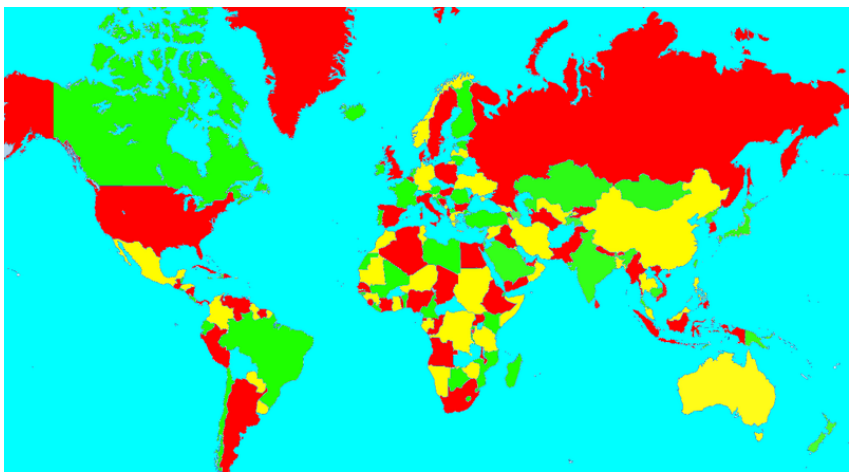
Un peu d'histoire (2)

Un des premiers problèmes résolus par la théorie des graphes
Pendant la bataille d'Angleterre
Constitution d'équipage (pilote/mécanicien) sur les avions

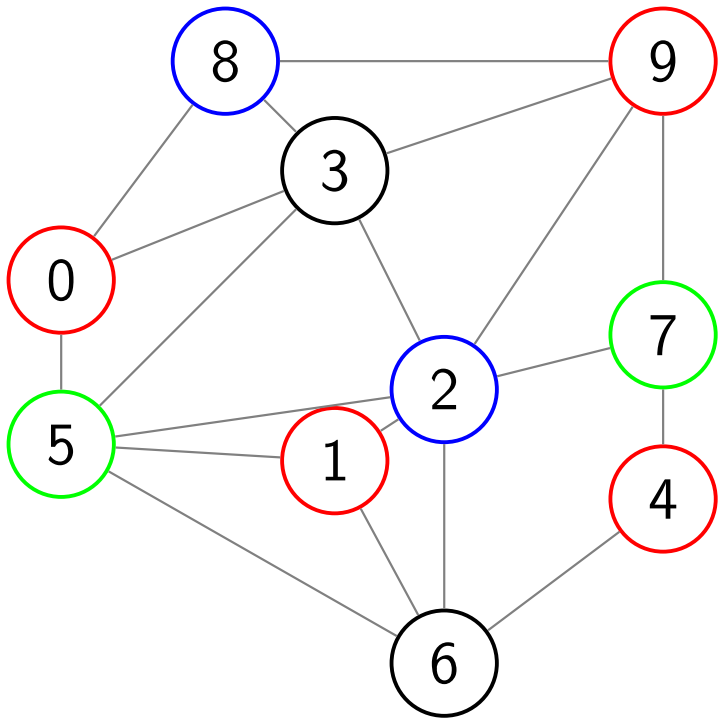


Un peu d'histoire (3)

Théorème des quatre couleurs ≡ 4-coloration des graphes planaires



Source : wikipedia



Type de relations modélisées

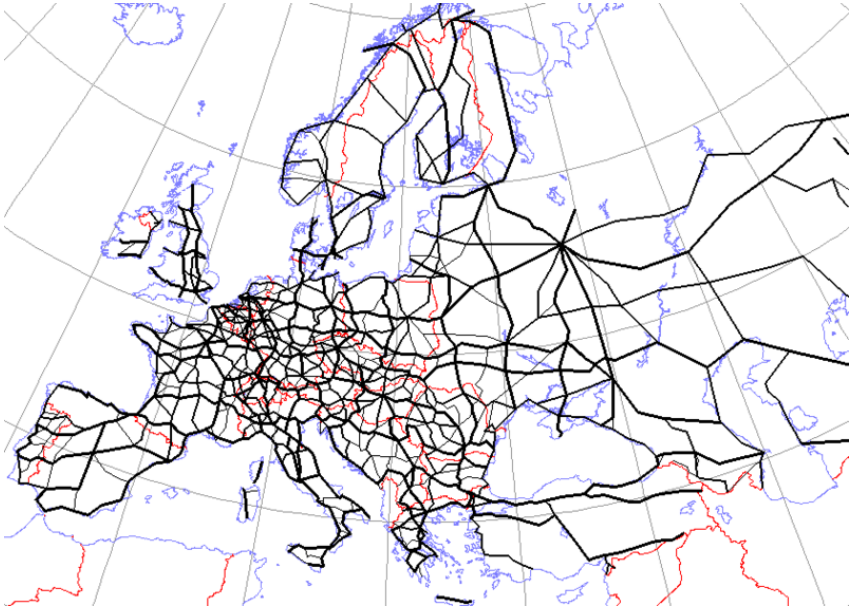
Les graphes représentent des relations binaires.

- réseaux (connexions, proximité, distance)
- disjonctions (compatible / incompatible)
- précédences / préférences
- inclusions, structures récursives (arbres, ...)
- matrices $n \times n$

Réseaux

- réseaux routiers
- réseaux telecom
- réseaux espace / temps
- réseaux sociaux, connaissances
- réseaux logiques (précédences, préférences)

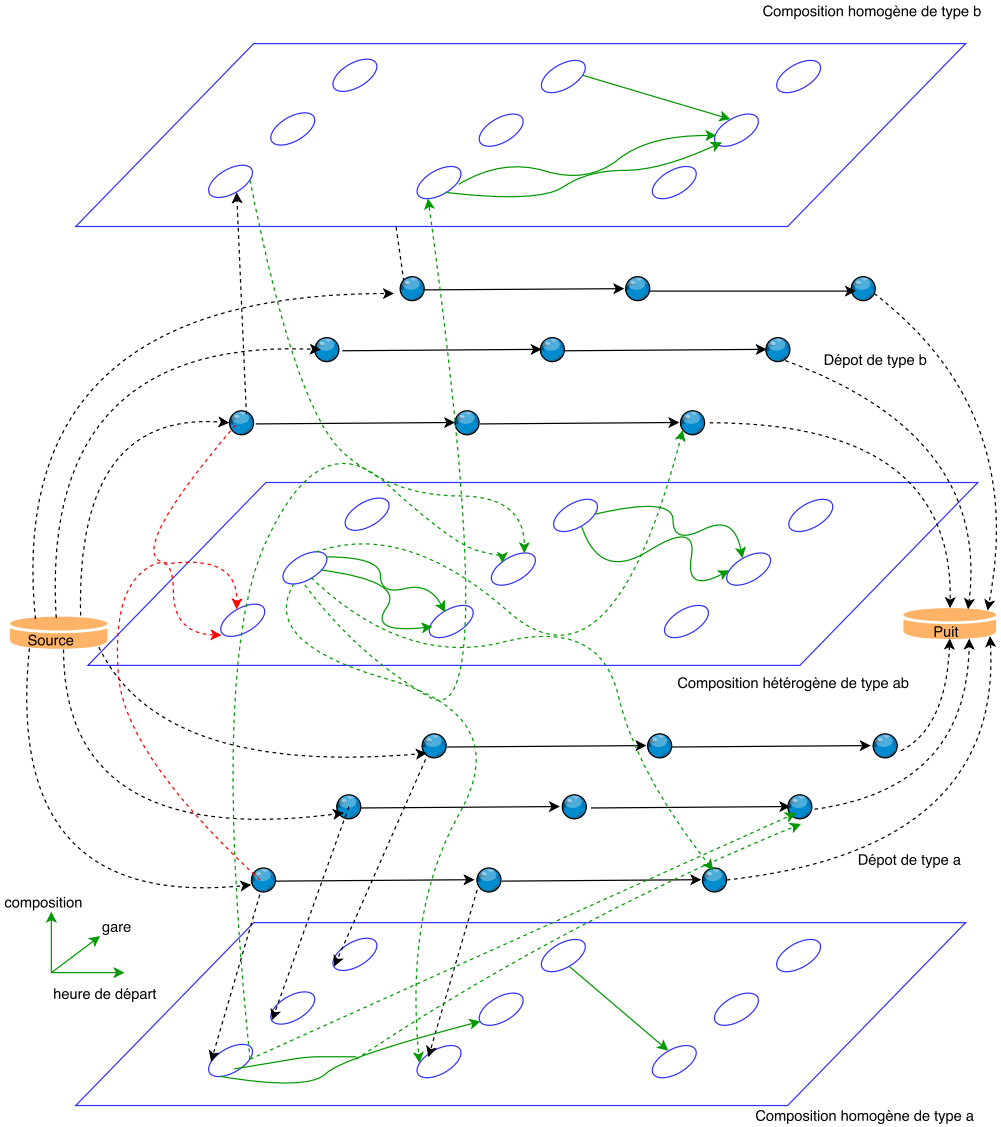
Réseaux



Source : wikipedia

Source : wikipedia

Réseau espace/temps/configuration pour le ferroviaire



Disjonctions

- incompatibilités (personnes, produits chimiques, ressources)
- compatibilités (couplages)
- emplois du temps
- problèmes d'affectation

Optimisation dans les graphes

Types de décisions à optimiser dans les graphes

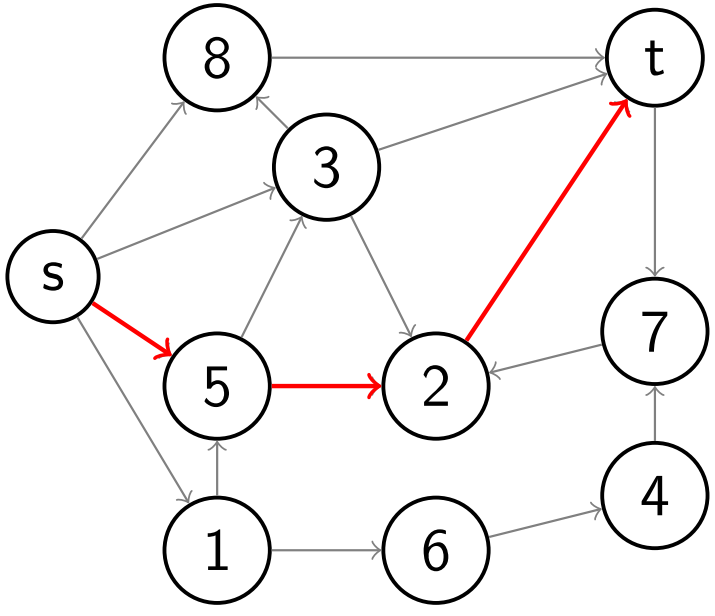
- recherche de chemins / circuits / cycles
- sélection d'un sous-ensemble de sommets
- calculer des partitions / couverture des sommets, des arcs
- affecter des valeurs aux arcs, aux sommets

Problème de plus court chemin dans un graphe

Trouver un chemin entre deux sommets qui minimise la valeur totale des arcs utilisés.

Plusieurs variantes en fonction des propriétés du graphe :

- Bellman : graphes sans circuits, $\Theta(n+m)$
- Dijkstra : graphes avec valuations positives, $\Theta(m \log n)$
- Bellman-Ford : graphes quelconques : $\Theta(nm)$



Complexité polynomiale : $\Theta(nm)$

Applications : route la plus rapide, la plus sûre, la plus rentable, ..., *résolution de problèmes formulés par une récurrence*

Modèle linéaire : plus court chemin entre deux sommets

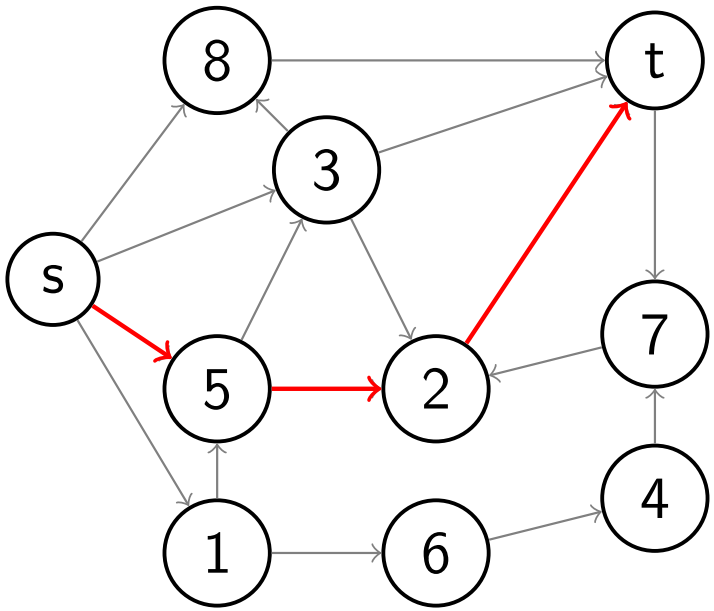
Variables

$$x_{ij} = \begin{cases} 1 & \text{si on choisit l'arc } (i,j) \\ 0 & \text{sinon} \end{cases}$$

Modèle

$$\min \sum_{(i,j)\in A} c_{ij}\, x_{ij}$$

$$\sum_{j\in N} x_{ij} - \sum_{j\in N} x_{ji} = \begin{cases} +1 & i = s \\ 0 & i \neq s,t \\ -1 & i = t \end{cases}$$



Propriété : la relaxation linéaire a une solution entière.

Arbre des plus courts chemins à partir d'une source

Variables

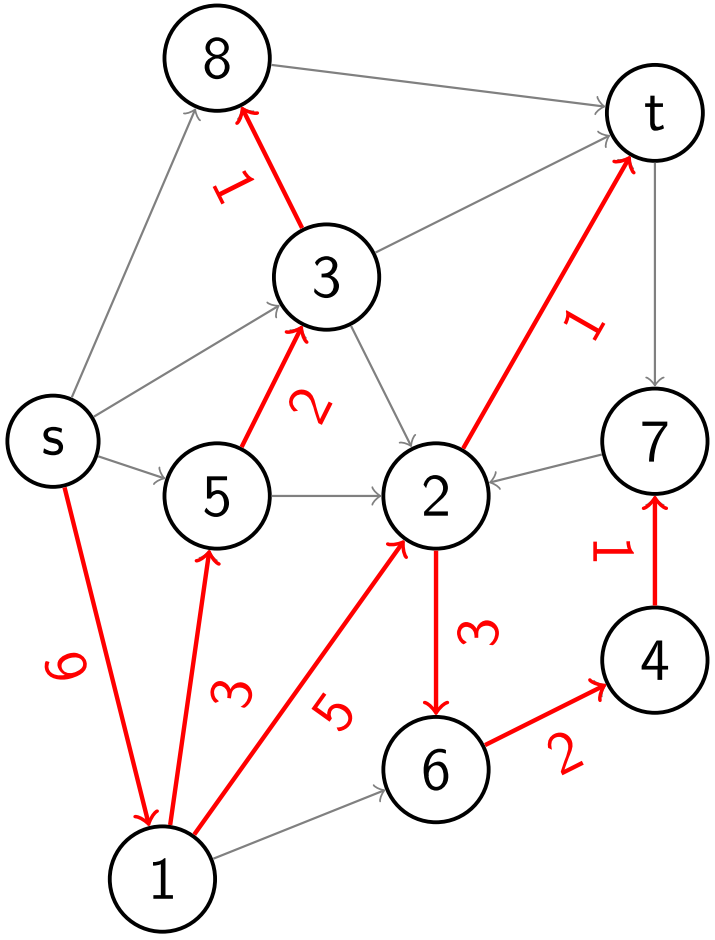
x_{ij} : nombre de chemins où l'arc (i,j) est choisi.

Modèle

$$\min \sum_{(i,j)\in A} c_{ij}\, x_{ij}$$

$$\sum_{j\in N} x_{ij} - \sum_{j\in N} x_{ji} = \begin{cases} n-1 & i = s \\ -1 & i \neq s \end{cases}$$

$$x_{ij} \in \mathbb{N} \text{ pour tout } (i,j)$$



Variantes difficiles

Tous les problèmes de cheminement ne sont pas faciles.

Plus court chemin élémentaire (avec arcs de coûts négatif)

Problème NP-difficile
Application : sous-problème dans la résolution de problèmes difficiles, collecte et paiement

Plus court chemin avec contraintes de ressources

Problème NP-difficile
Application : consommation de ressources, manière de traiter l'élémentarité, distance ou temps maximum, etc.

Méthodes utilisées : programmation dynamique, heuristiques

Circuit / cycle, chemin/chaîne eulérien(ne)

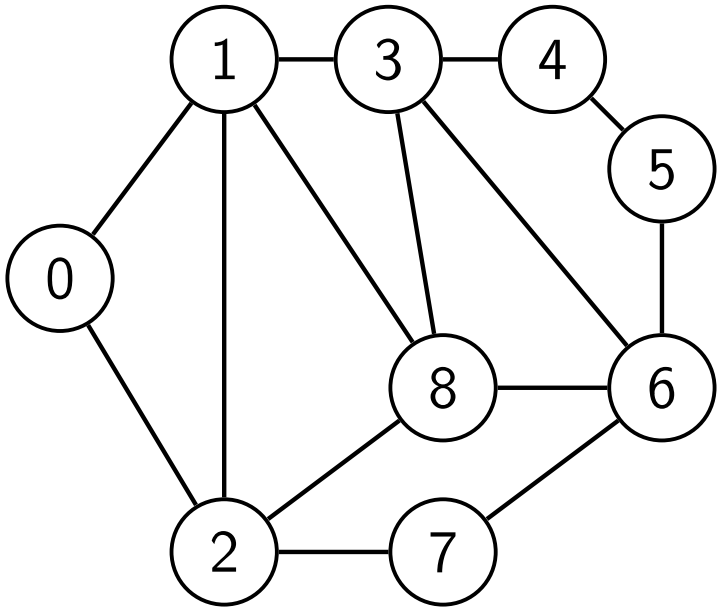
Un chemin (chaîne) eulérien(ne) est un chemin qui emprunte une fois et une seule chaque arc (arête) du graphe.

Proposition

Un graphe admet un cycle eulérien si et seulement si il est connexe et tous ses sommets sont de degré pair.

Applications : inspection des routes, de câbles, de canalisations

Complexité polynomiale : $\Theta(m)$



Circuit / cycle, chemin/chaîne eulérien(ne)

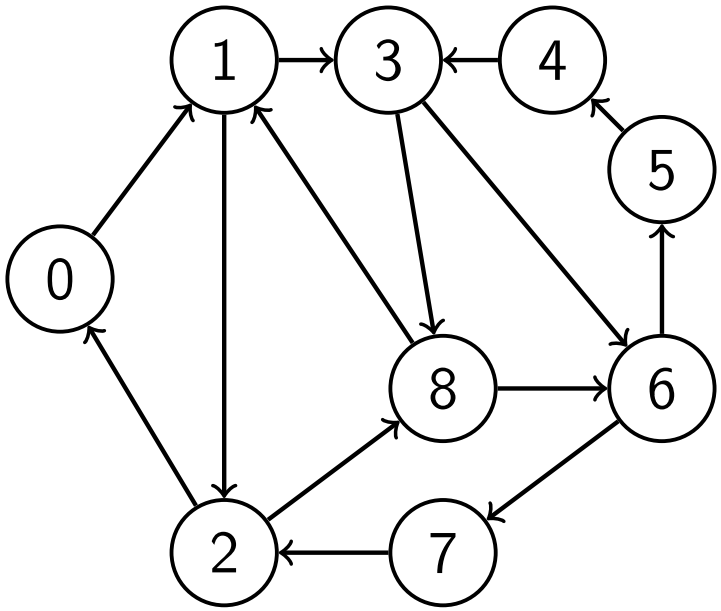
Un chemin (chaîne) eulérien(ne) est un chemin qui emprunte une fois et une seule chaque arc (arête) du graphe.

Proposition

Un graphe admet un circuit eulérien si et seulement si il est fortement connexe et chaque sommet v est tel que $deg^-(v) = deg^+(v)$.

Applications : inspection des routes, de câbles, de canalisations

Complexité polynomiale : $\Theta(m)$



Circuit / cycle, chemin/chaîne eulérien(ne)

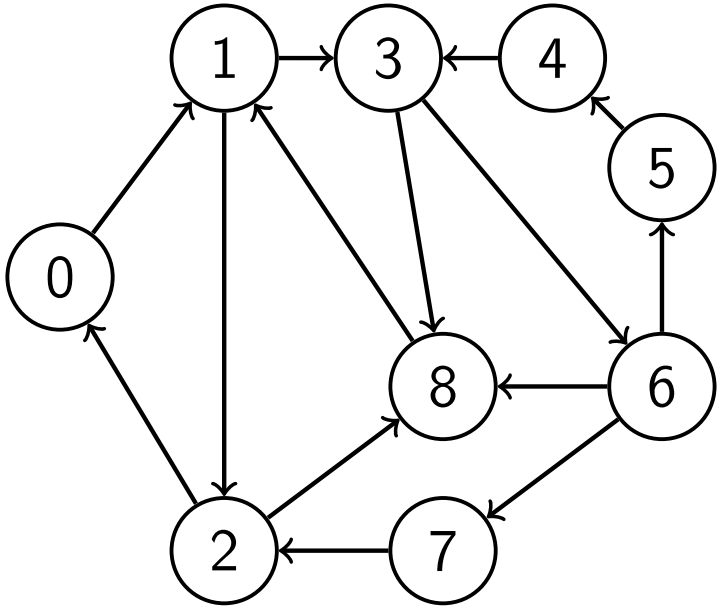
Un chemin (chaîne) eulérien(ne) est un chemin qui emprunte une fois et une seule chaque arc (arête) du graphe.

Proposition

Un graphe admet un circuit eulérien si et seulement si il est fortement connexe et chaque sommet v est tel que $deg^-(v) = deg^+(v)$.

Applications : inspection des routes, de câbles, de canalisations

Complexité polynomiale : $\Theta(m)$

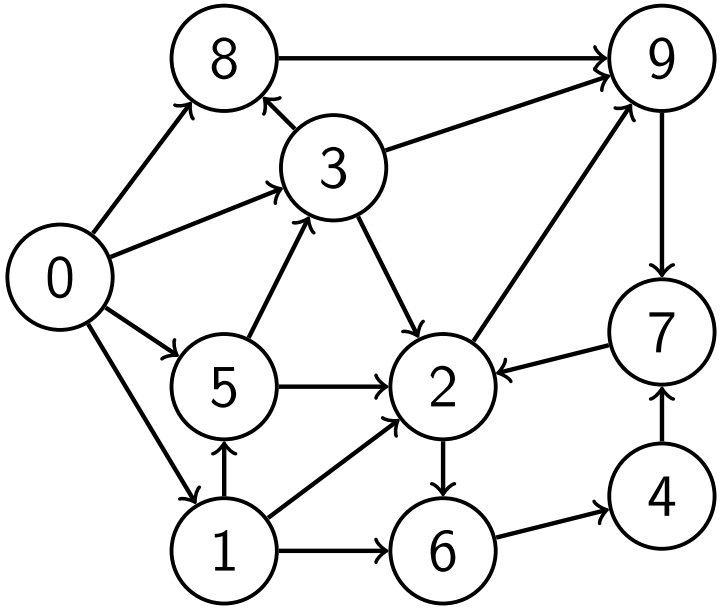


Chemin/circuit hamiltonien

Un chemin (chaîne) hamiltonien(ne) est un chemin (chaîne) à qui emprunte une fois et une seule chaque sommet du graphe.

Applications : inspection d'installations, voyageur de commerce, ...

NP-difficile.



Problèmes de flot

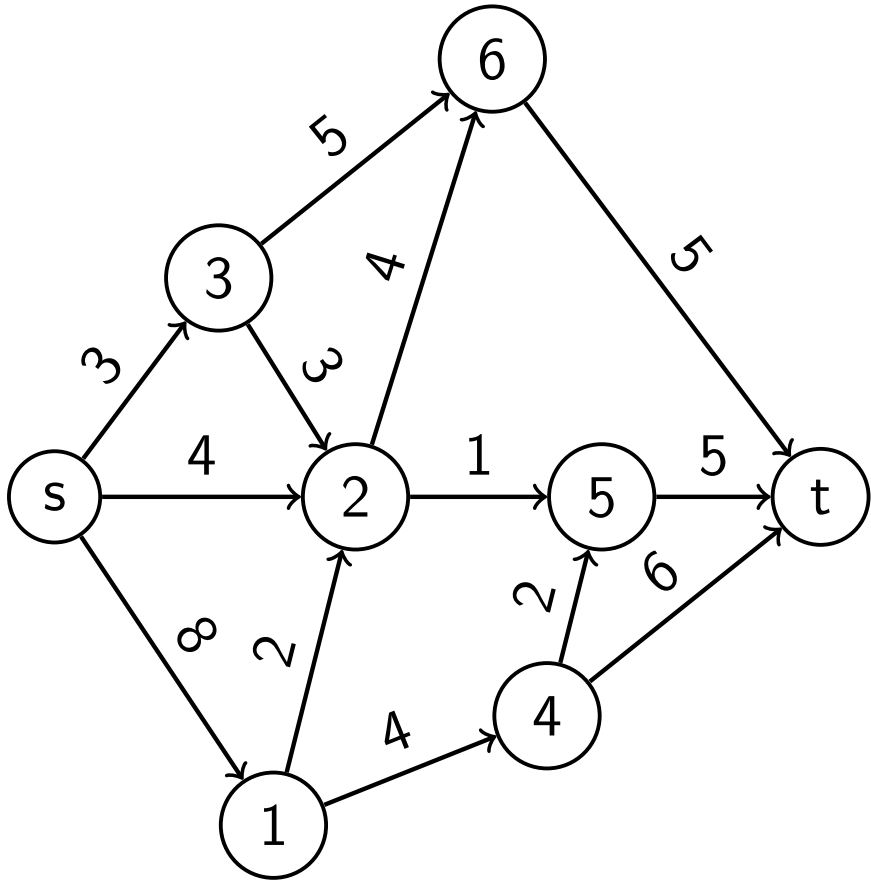
Definition

Un **réseau de transport** est un graphe $G = (V, A, c)$ valué positivement, avec une racine s, et un puit t.

Definition

Un **flot** est une application de $f : A \mapsto \mathbb{R}$ qui vérifie

- contraintes de capacité : $0 \leq f_{ij} \leq c_{ij}$
- contraintes de Kirchoff : $\forall i \in V \setminus \{s, t\}, \sum_{j \in N^+(i)} f_{ij} = \sum_{k \in N^-(i)} f_{ki}$



Applications : flux de données, de véhicules, de liquide, ...

Flot max : Complexité polynomiale : $O(n^3)$

Problèmes de flot

Definition

Un **réseau de transport** est un graphe $G = (V, A, c)$ valué positivement, avec une racine s, et un puit t.

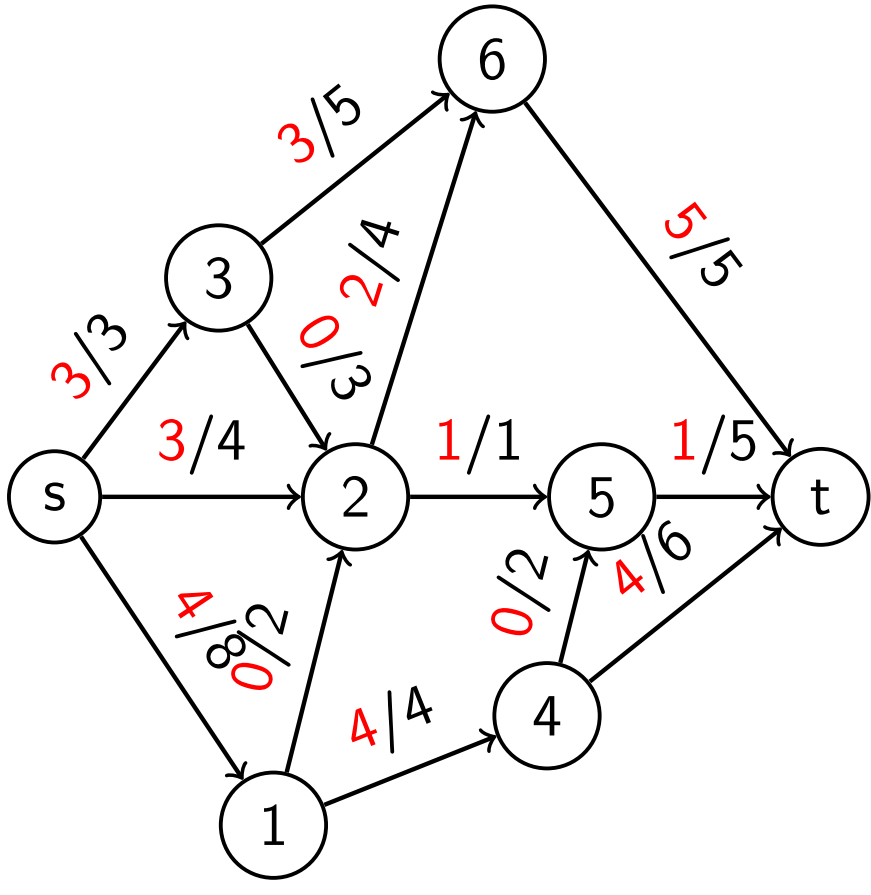
Definition

Un **flot** est une application de $f : A \mapsto \mathbb{R}$ qui vérifie

- contraintes de capacité : $0 \leq f_{ij} \leq c_{ij}$
- contraintes de Kirchoff : $\forall i \in V \setminus \{s, t\}, \sum_{j \in N^+(i)} f_{ij} = \sum_{k \in N^-(i)} f_{ki}$

Applications : flux de données, de véhicules, de liquide, ...

Flot max : Complexité polynomiale : $O(n^3)$



Modèles de flot dans des réseaux

RÉSEAU	NOEUDS	ARCS	FLOT
Électrique	stations de géné.	lignes élec.	électricité
Téléphone	sta. de contrôle	câble	communications
Web	serveurs	câbles	bits
Usine	machines	tapis roulant	produits
Routier	carrefours	rue	véhicules, biens
Espace/temps	endroit/heure	dist./temps	personnel

Flot max / coupe min

Définition

Une **coupe** est une partition des sommets d'un graphe en deux parties.

Définition alternative

On désigne parfois par coupe les arcs ayant une extrémité dans chaque sous-ensemble de la partition.

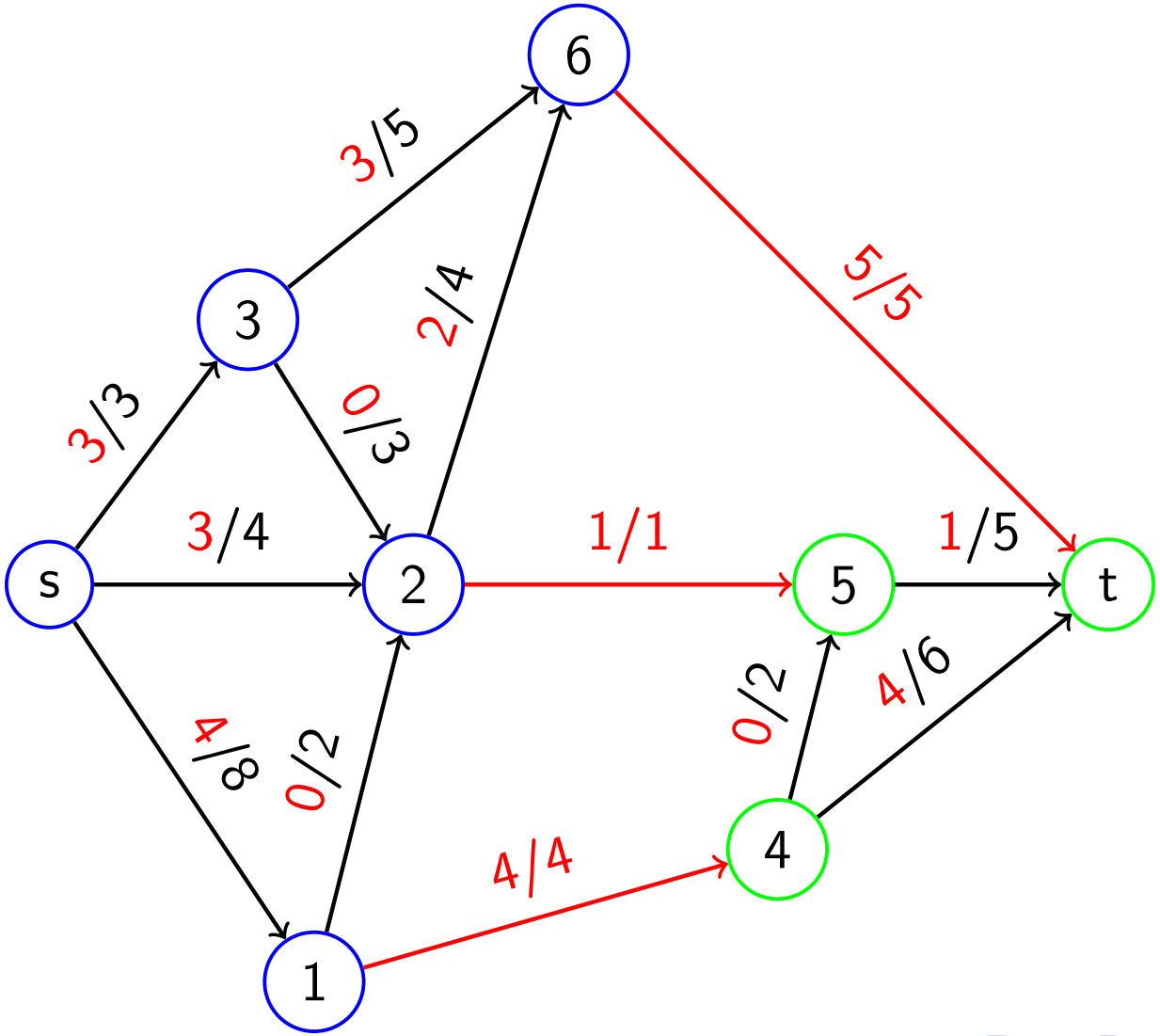
Définition

La **capacité d'une coupe** est la somme des capacités des arcs de la coupe.

Théorème

La capacité minimale d'une coupe est égale au flot maximum.

Flot max / coupe min



Flot maximum : programme linéaire

Modèle linéaire

$$\max \quad v$$

$$\sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = \begin{cases} v & \text{pour } i = s \\ 0 & \text{pour } i \neq s, t \\ -v & \text{pour } i = t \end{cases}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad \text{pour tout } (i,j) \in A$$

Note : la matrice de contraintes est unimodulaire (= solution entière si les u_{ij} sont entiers).

Flot de coût minimum : programme linéaire

On utilise un paramètre v qui est la valeur du flot souhaité (par exemple celle du flot max).

Modèle

$$\min \sum_{(i,j)\in A} c_{ij}\, x_{ij} \tag{1}$$

$$\sum_{j\in N} x_{ij} - \sum_{j\in N} x_{ji} = \begin{cases} v & \text{pour } i = s \\ 0 & \text{pour } i \neq s,t \\ -v & \text{pour } i = t \end{cases}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad \text{pour tout } (i,j) \in A$$

Les flots sont partout...

Quelques collaborations industrielles de membres de l'équipe RealOpt (Inria Bordeaux Sud-Ouest) depuis 10 ans

- SNCF : modèles de flot dans les hypergraphes pour la planification des trains
- St Gobain : modèles de flot dans les hypergraphes pour la découpe de verre
- EDF : modèles de flot dans les graphes pour la planification des arrêts de tranche nucléaires
- Ertus ; modèles de flot dans les graphes pour la planification de produits phytosanitaires
- Vekia : modèles de flot dans les graphes pour la planification des employés dans la vente au détail
- Opalean : modèles de flot pour optimiser l'échange de palettes

Clique maximum

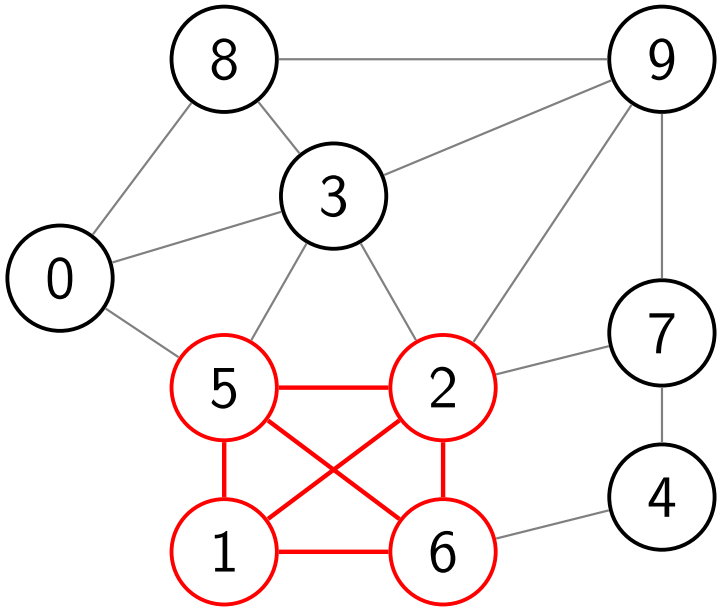
Problème

Déterminer une clique de cardinalité maximum.

Applications : affectation, analyse de réseaux sociaux, bioinformatique, chimie

NP-difficile dans le cas général.

Bien résolu en pratique sur des instances arbitraires.



Variantes

Clique de poids maximum, lister toutes les cliques maximales.

Ensemble stable maximum

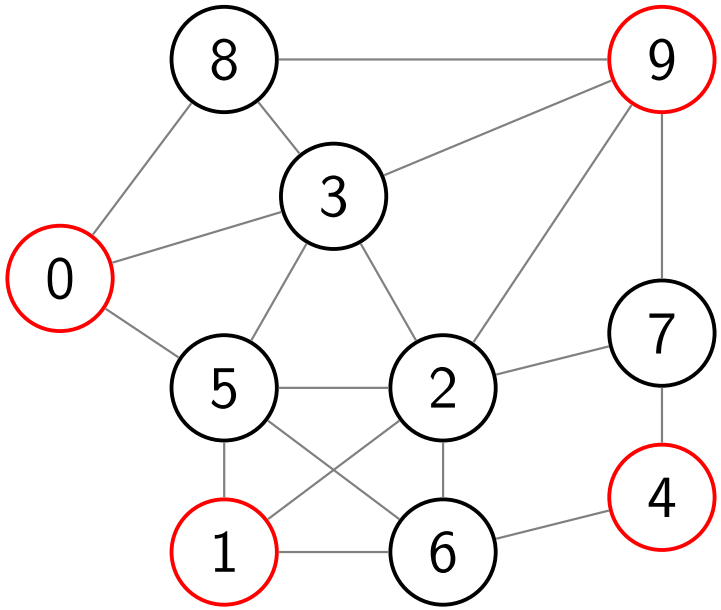
Definition

Un ensemble stable est un ensemble de sommets S tel que $E \cup (S \times S) = \emptyset$.

Problème du stable de cardinalité maximum

Déterminer un ensemble indépendant de de cardinalité maximum.

NP-difficile dans le cas général.



Coloration de cardinalité minimum

Definition

Une **coloration propre** pour un graphe G est une affectation de couleurs aux sommets de G de telle manière que deux sommets adjacents ne soient pas de la même couleur.

Définition alternative

Une coloration d'un graphe $G = (V, E)$ est une partition de V en ensembles stables.

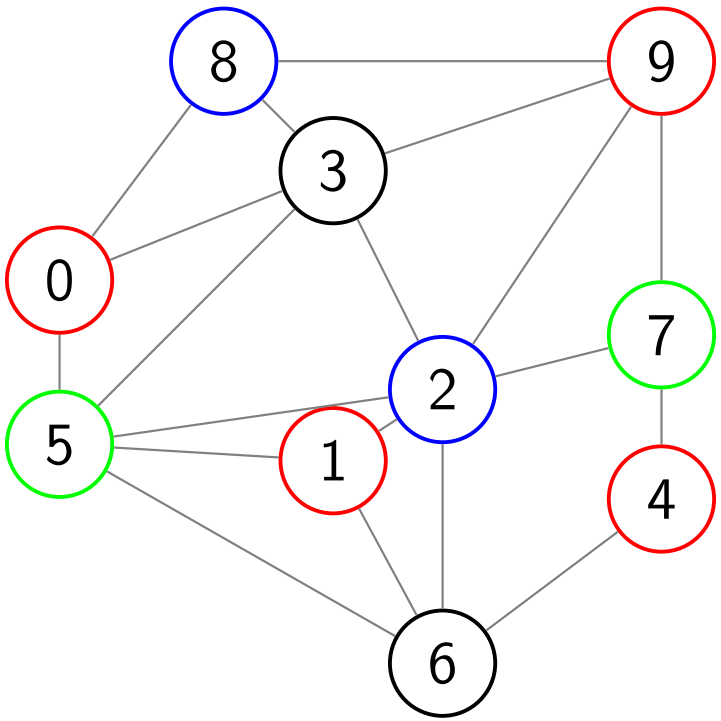
Coloration de cardinalité minimum

On appelle **nombre chromatique** χ d'un graphe G le nombre minimum de couleurs nécessaires pour colorer G.

Applications : affectation, ordonnancement, allocation de fréquences, allocation de registres,

NP-difficile dans le cas général.

Coloration de cardinalité minimum



Problème d'arbre couvrant de poids minimum

Définition

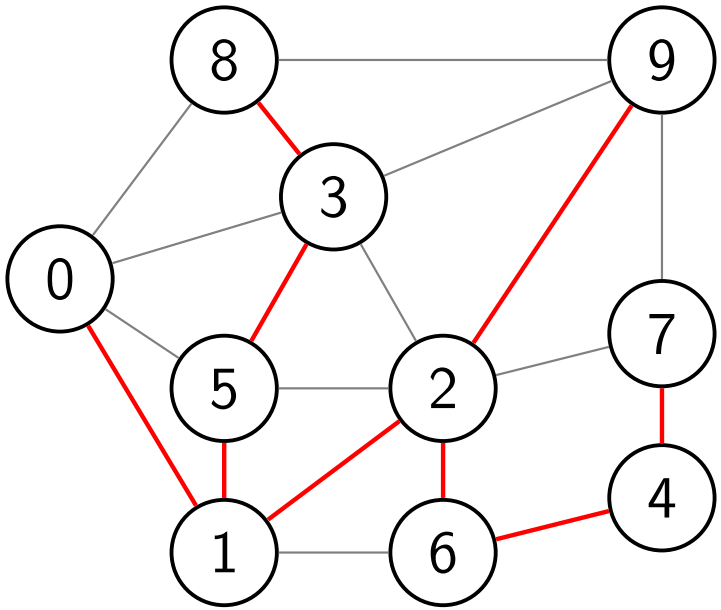
Un **arbre couvrant** d'un graphe $G = (V, E)$ est un arbre $T = (V, E')$ t.q. $E' \subseteq E$.

Arbre couvrant de poids minimum

Parmi tous les arbres couvrant un graphe G, en déterminer un de poids minimum.

Applications : mise en place d'un réseau télécom', analyse de données, segmentation d'images, ...

Complexité polynomiale : $\Theta(m \log n)$



Modélisation PLNE

Modèle

$$\begin{aligned} \min \quad & \sum_{e \in E} w_e x_e \\ \text{s.t.} \quad & \sum_{e \in E} x_e = n-1 \\ & \sum_{e \in E \cap R \times R} x_e \leq |R|-1, \quad \forall R \subseteq V \\ & x_e \in \{0,1\}, \quad \forall e \in E \end{aligned}$$

Variables : $\forall e \in E$, $x_e = \begin{cases} 1 \text{ si l'arc } e \text{ appartient à l'arbre,} \\ 0 \text{ sinon.} \end{cases}$

Contraintes :

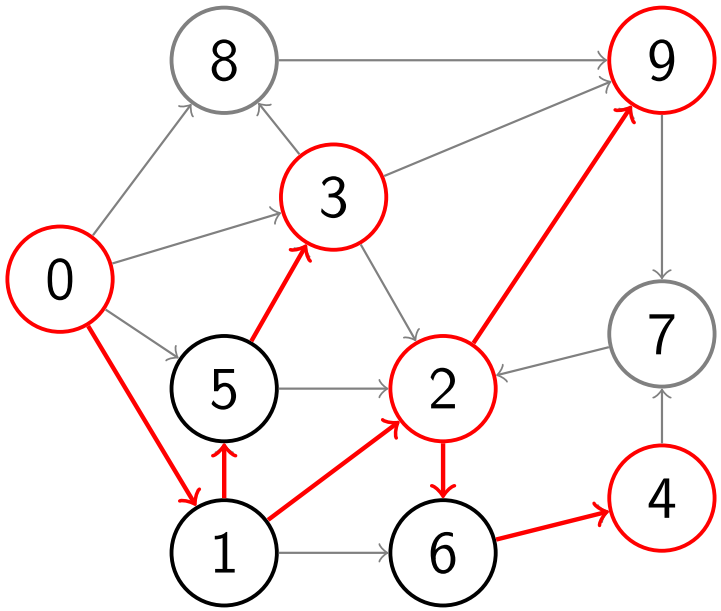
- on choisit $n-1$ arêtes
- dans chaque sous-graphe de taille k, on a au plus $k-1$ arêtes

Arbre de Steiner

Soit $G = (V, E, u)$ un graphe valué, et $S \subseteq V$ un sous-ensemble de sommets de ce graphe.

Definition

Le problème du calcul d'un **arbre de Steiner** consiste à déterminer l'arbre $T = (S', E')$ tel que $S \subseteq S' \subseteq V$ et $E' \subseteq E$ qui minimise la somme des valuations sur les arêtes.



Applications : mise en place d'un réseau télécom', analyse de données, segmentation d'images, ...

NP-difficile.

Formulation PLNE

Modèle (cas valuations positives)

$$\begin{array}{lll} \min & \sum_{e \in E} w_e x_e & \\ \text{s.t.} & \sum_{e \in \delta(R)} x_e \geq 1, & \forall R \subset V, R \cap S \neq \emptyset, (V \setminus R) \cap S \neq \emptyset \\ & x_e \in \{0,1\}, & \forall e \in E \end{array}$$

avec $\delta(R)$: ensemble d'arcs ayant une extrémité dans R et une dans $V \setminus R$.

En clair : pour toute coupe séparant deux sommets de S, on doit sélectionner une arête sortant de cette coupe

Ensemble dominant

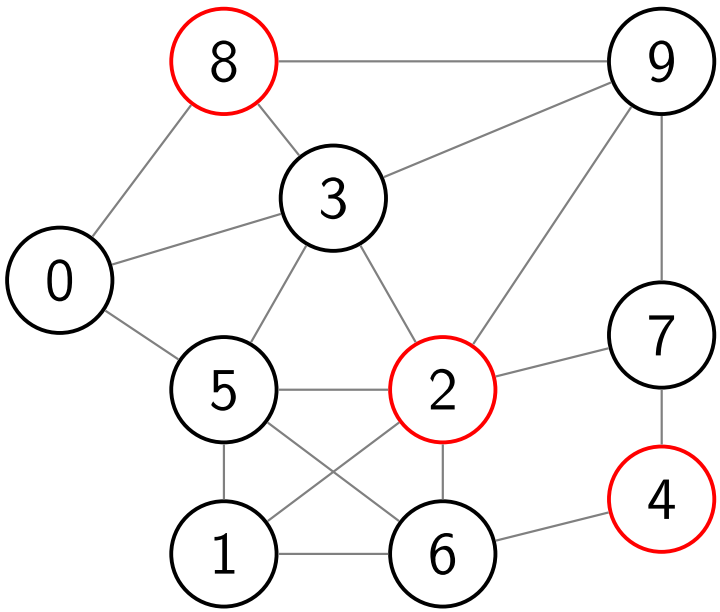
Definition

Un **ensemble dominant** d'un graphe $G = (V, E)$ est un sous-ensemble W de V tel que tout sommet de $V \setminus W$ est adjacent à un sommet de V.

Problème

Déterminer un ensemble dominant de cardinalité minimum.

NP-difficile dans le cas général.



Couverture par sommets

Couverture

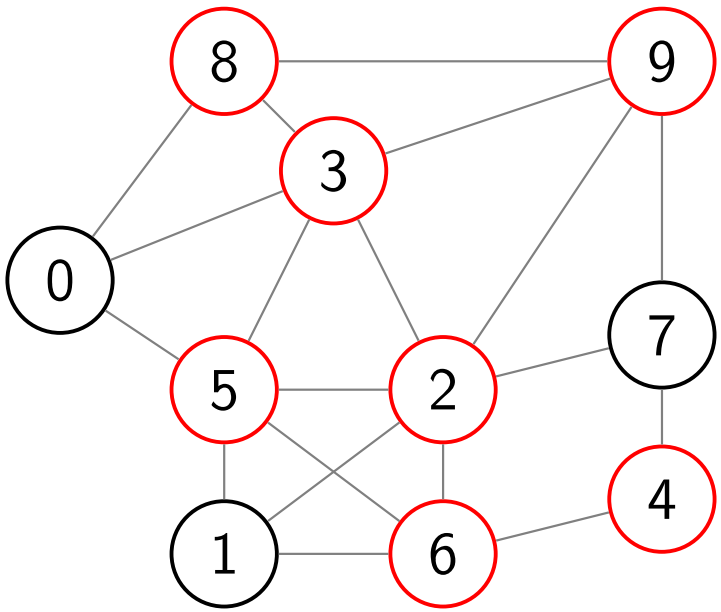
Une couverture par sommets d'un graphe $G = (V, E)$ est un ensemble de sommets $W \subseteq V$ t.q. toute arête de E a une extrémité dans W.

Problème

Déterminer une couverture de cardinalité minimum.

Applications : affectation de capteurs, ...

NP-difficile dans le cas général.



Couplage

Couplage

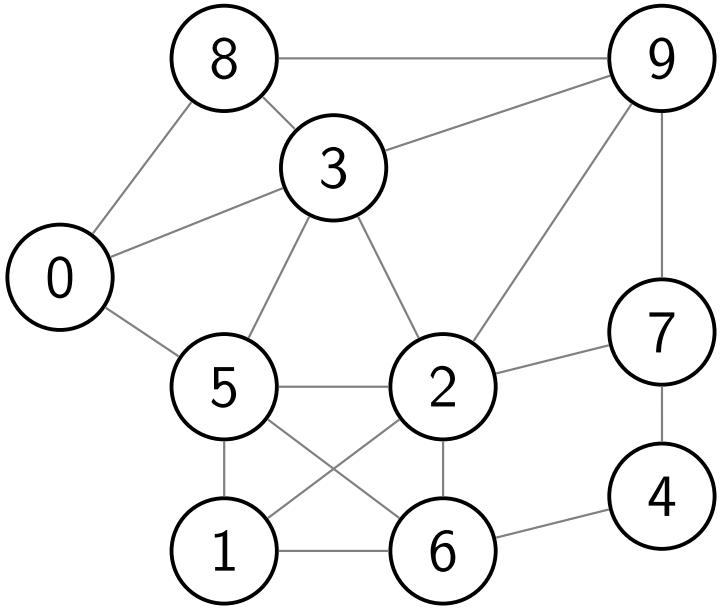
Un couplage est un ensemble d'arête deux à deux non incidentes.

Couplage de cardinalité maximum

Déterminer un couplage de cardinalité maximum.

Applications : couplage d'employés, détection radar

Complexité polynomiale : $O(m\sqrt{n})$



Couplage

Couplage

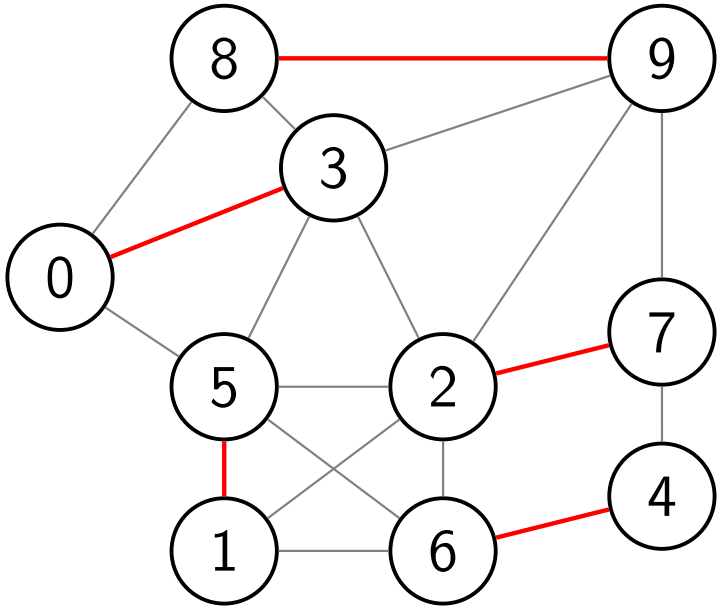
Un couplage est un ensemble d'arête deux à deux non incidentes.

Couplage de cardinalité maximum

Déterminer un couplage de cardinalité maximum.

Applications : couplage d'employés, détection radar

Complexité polynomiale : $O(m\sqrt{n})$



Problèmes classiques : bilan

Problème	complexité
connexité	$\Theta(n+m)$
plus court chemin	$\Theta(n*m)$
flot max	$\Theta(n^3)$
eulérien	$O(n+m)$
hamiltonien	NP-difficile
clique max	NP-difficile
coloration	NP-difficile
ensemble dominant	NP-difficile
couverture par sommet	NP-difficile
couplage	$\Theta(m\sqrt{n})$
arbre couvrant	$\Theta(m \log n)$
arbre de Steiner	NP-difficile

A retenir

Parmi les problèmes de graphes qui se posent en pratique, beaucoup sont théoriquement difficiles

Attention néanmoins de ne pas rater les cas particuliers faciles.

Dans le cas où les problèmes sont difficiles :

- heuristiques
- programmes mathématiques de grande taille (à générer dynamiquement)

Le problème du Voyageur de Commerce

Problème :

- un voyageur de commerce habitant dans la ville 1 doit visiter n clients habitant chacun dans la ville $i = 2, \ldots, n$.
- On connait les distances d_{ij} entre chaque paire de villes (i, j), $i, j = 1, \ldots, n$.
- Le problème consiste à trouver la séquence (l'ordre) dans laquelle il doit visiter les clients de manière à minimiser la distance totale.

Applications

Nombreuses applications directes.

- Tournées d'inspection de matériel
- Livraison
- Collecte
- Drones
- Perçage de cartes à puces
- Planification d'opérations
- ...

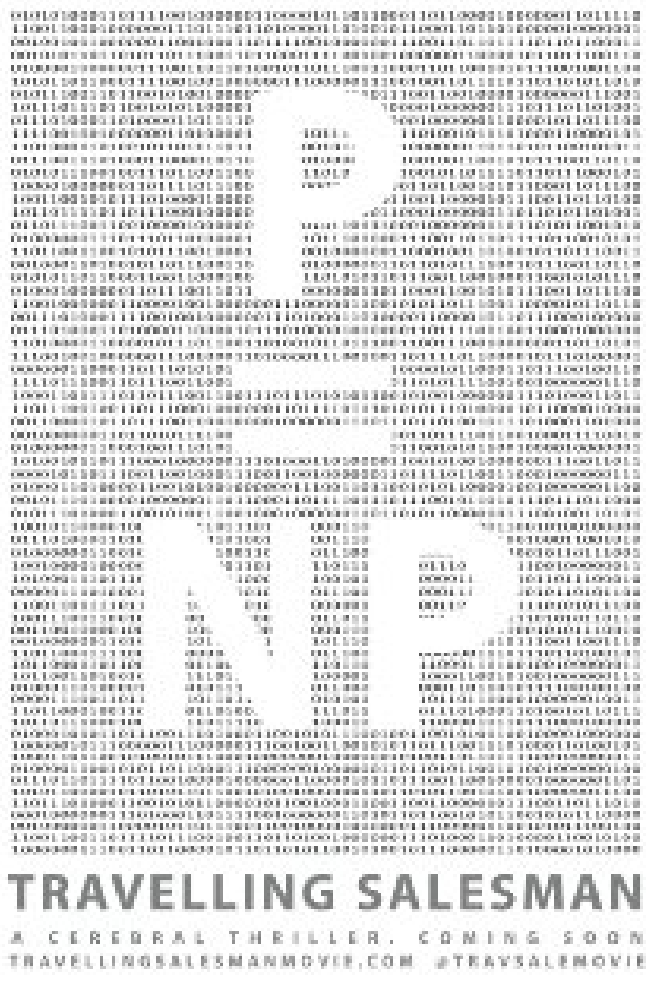
Existe aussi comme sous-problème
d'autres applications.

Applications

Nombreuses applications directes.

- Tournées d'inspection de matériel
- Livraison
- Collecte
- Drones
- Perçage de cartes à puces
- Planification d'opérations
- ...

Existe aussi comme sous-problème d'autres applications.



Variantes du problème

Il existe plusieurs versions, extensions et cas particuliers du problèmes de voyageur de commerce.

- Symétrique / asymétrique
- Métrique (inégalité triangulaire respectée)
- Fenêtres de temps
- Distances dépendant du temps
- Possibilité de ne pas visiter des clients (en payant une pénalité)
- Visiter tous les arcs (postier chinois)

Complexité

Proposition

Le problème de voyageur de commerce est NP-difficile (par réduction de cycle hamiltonien).

Enumération : Chaque permutation des numéros de villes allant de 2 à n définit un circuit possible :

$$\pi = (1, \pi_2, \ldots, \pi_n)$$

où $\pi_k = i$ indique que la ville i est en position k.

Nombre de circuits possibles $= \frac{(n-1)!}{2}$.

Énumération de toutes les possibilités

Enumération de toutes les solutions sur un ordinateur 4 GHz.
Un tel ordinateur effectue 4 milliards d'opérations à la seconde.
Supposons qu'il évalue 4 milliards (4.10^9) circuits à la seconde.

- Pour 10 villes ($n = 10$) :
  Nombre de circuits $= 181\,440 \approx 2.10^6$.
  Temps d'évaluation $=$ 0.5 millièmes de seconde.
- Pour 30 villes ($n = 30$) :
  Nombre de circuits $\approx 44.10^{29}$

  Temps d'évaluation $\approx 1.10^{21}$ secondes
  $\approx 3.10^{17}$ heures
  $\approx 13.10^{15}$ jours
  $\approx 35.10^{12}$ années
  $\approx 35\,000$ milliards d'années

Énumération de toutes les possibilités

Enumération de toutes les solutions sur un ordinateur 4 GHz.
Un tel ordinateur effectue 4 milliards d'opérations à la seconde.
Supposons qu'il évalue 4 milliards (4.10^9) circuits à la seconde.

- Pour 10 villes ($n = 10$) :
  Nombre de circuits $= 181\,440 \approx 2.10^6$.
  Temps d'évaluation $=$ 0.5 millièmes de seconde.
- Pour 30 villes ($n = 30$) :
  Nombre de circuits $\approx 44.10^{29}$

  Temps d'évaluation $\approx 1.10^{21}$ secondes
  $\approx 3.10^{17}$ heures
  $\approx 13.10^{15}$ jours
  $\approx 35.10^{12}$ années
  $\approx 35\,000$ milliards d'années

En optimisation combinatoire, big data = 30 éléments !

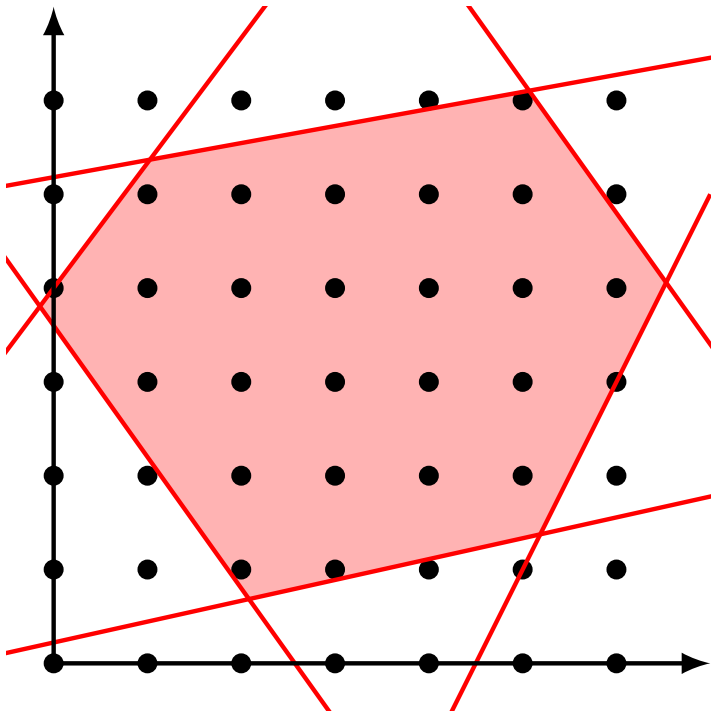
Programmation linéaire en nombres entiers

On peut écrire ainsi un programme linéaire avec n variables $x_1, \ldots, x_n$ et m contraintes.

$$\begin{array}{rc} \max & \sum_{i=1}^{n} c_i x_i \\ \text{sous les contraintes} & \sum_{i=1}^{n} a_{ij} x_i \leq b_j, (j = 1, \ldots, m) \\ & x_i \in \mathbb{Z}, (i = 1, \ldots, n) \end{array}$$

Les solveurs modernes sont capables de résoudre des problèmes de grande taille... s'ils ont de bonnes propriétés.

Qualité d'une formulation



Modèle linéaire en nombres entiers

Variables : $x_{ij} \in \{0, 1\}$

Objectif : minimiser la distance totale : $\sum_{(i,j)\in A} c_{ij} x_{ij}$

Contraintes :

- on entre une fois dans chaque sommet : $\forall i \in V, \sum_{j\in V} x_{ij} = 1$
- on sort une fois de chaque sommet : $\forall i \in V, \sum_{j\in V} x_{ji} = 1$

Problème : contraintes de sous-tours.

Deux méthodes classiques :

- Miller-Tucker-Zemlin (MTZ)
- Énumération des sous-tours

Formulations type Miller Tucker Zemlin (MTZ)

On numérote les sommets $1, \ldots, n$ pour éviter de revenir sur un sommet déjà visité.

On introduit pour cela des variables u_i pour $i = 1, \ldots, n$.

Formulation de Millet Tucker Zemlin

$u_1 = 1,$
$u_i - u_j + (n-1)x_{ij} \leq n-2, \ \forall i \neq 1, \ \forall j \neq 1.$

Version améliorée de Desrochers et Laporte

$u_i - u_j + (n-1)x_{ij} + (n-3)x_{ji} \leq n-2, \ \forall i \neq j = 2, \ldots, n$

Formulation "sous-tours"

Soit S l'ensemble de tous les sous-tours.

$\forall s \in S, \sum_{(i,j)\in s} x_{ij} \leq |s| - 1$

$\Longrightarrow$ De l'ordre de 2^n contraintes.

Solution : ne générer que les contraintes utiles (méthode itérative de résolution / génération de coupes).

Modèle obtenu

$$\min \sum_{(i,j)\in A} c_{ij} x_{ij}$$

$$\sum_{j\in V} x_{ij} = 1, \quad \forall i \in V$$

$$\sum_{j\in V} x_{ji} = 1, \quad \forall i \in V$$

$$\sum_{(i,j)\in s} x_{ij} \leq |s| - 1, \quad \forall s \in S$$

$$x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A$$

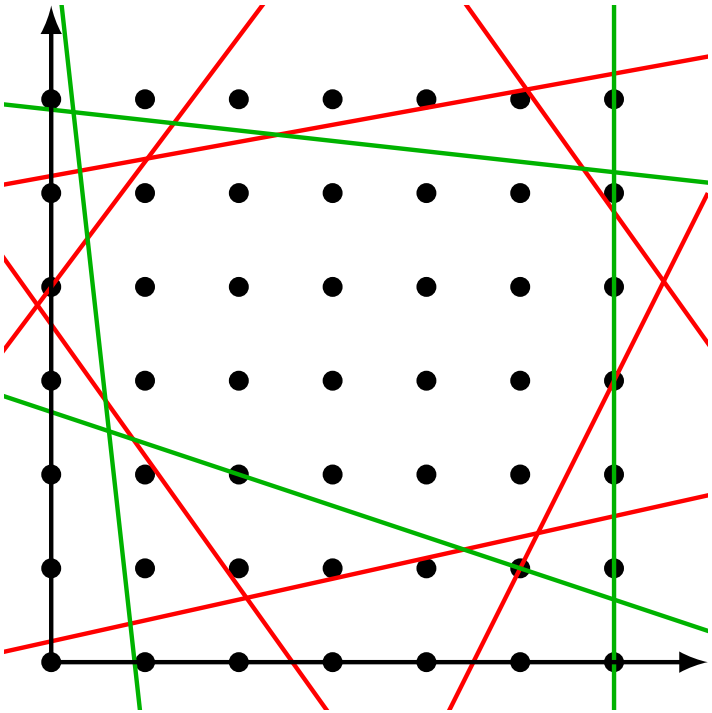
Fonctionnement d'un solveur de PLNE

Les solveurs PLNE reposent sur des algorithmes de branch-and-cut. Il s'agit d'énumérer toutes les solutions possibles en utilisant la relaxation linéaire pour stopper le traitement d'un noeud.

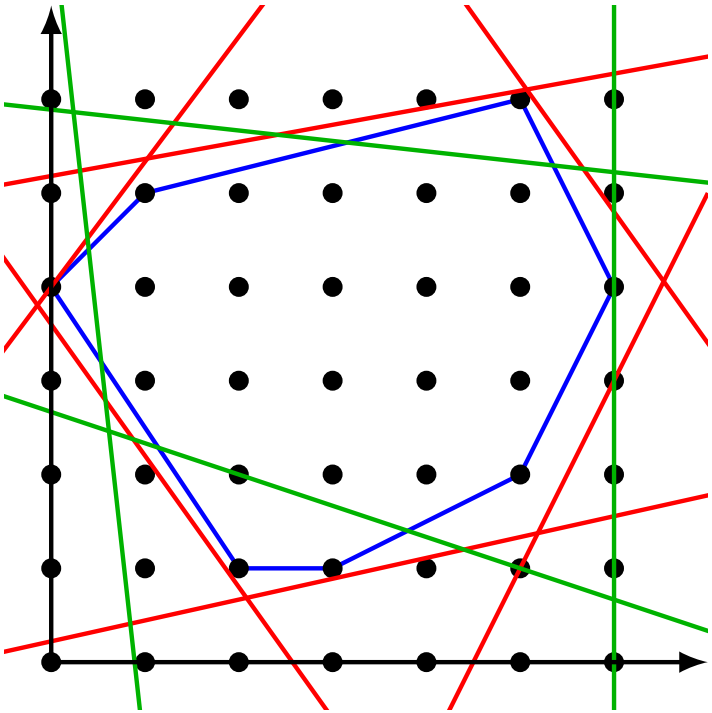
1. faire
   - 1.1 calculer la relaxation linéaire
   - 1.2 éliminer les variables grâce aux coûts réduits
   - 1.3 tester s'il existe des contraintes connues violées par la solution fractionnaire courante
   - 1.4 si oui, les ajouter
2. tant que des nouvelles coupes ont été ajoutées

Problème : comment savoir qu'une contrainte connue est violée par la solution courante ?

Génération dynamique de la formulation

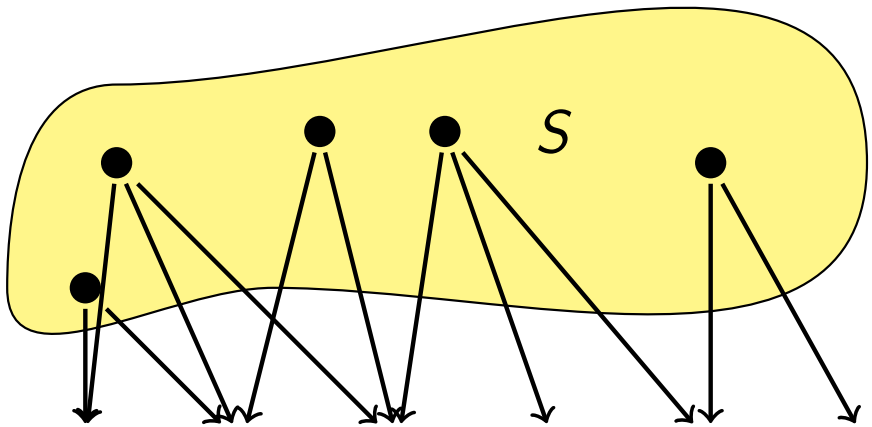


Génération dynamique de la formulation



Séparation des contraintes de sous-tours

On reformule la contrainte de la manière suivante.
$\forall S \subset V, \sum_{a \in \Gamma^+(S)} x_a \geq 1$
où $\Gamma^+(S)$ est l'ensemble des arcs (u, v) t.q. $u \in S$ et $v \in V \setminus S$.



Algorithme de séparation : coupe min = flot max

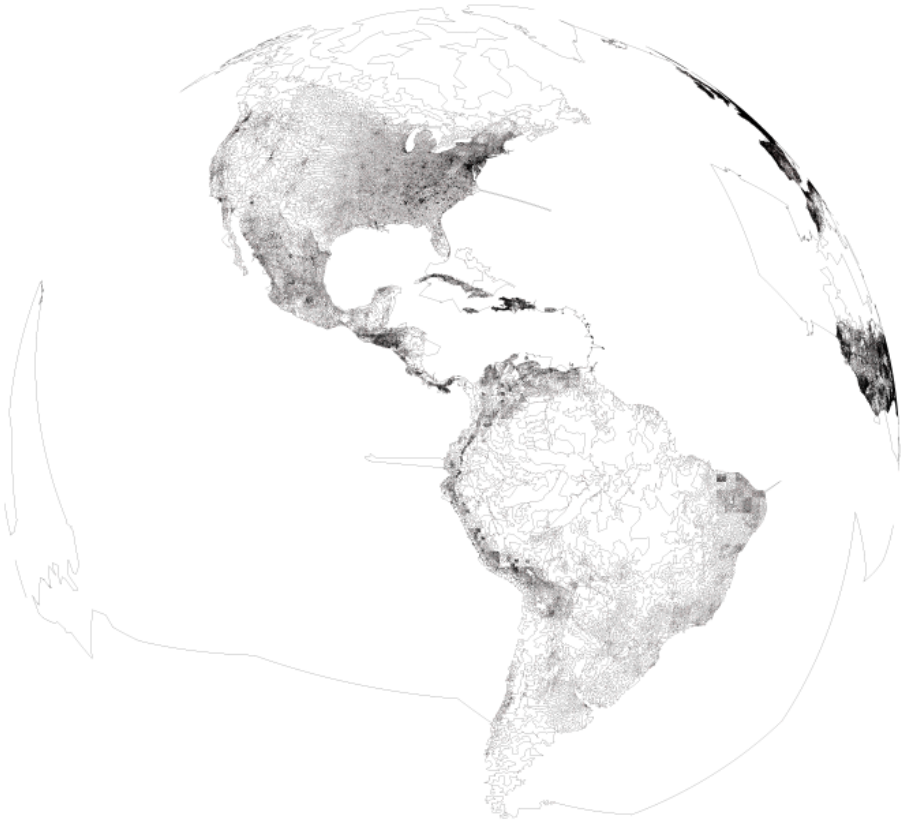
Meilleurs résultats pour les méthodes exactes

Les améliorations du modèle présenté permettent de résoudre un problème à 1,904,711 villes à 0,076% de précision.

Améliorations :

- Ajout des contraintes de sous-tours uniquement lorsque c'est nécessaire
- Ajout de nombreuses inégalités valides pour améliorer la formulation

Meilleurs résultats pour les méthodes exactes



A retenir pour cette partie

Même lorsque le problème est NP-difficile, il est possible de le résoudre à l'exact.

Moyennant un peu (souvent beaucoup) d'efforts.

Dans de nombreux cas, le décideur se satisfait d'une solution *"de bonne qualité"*.

Les solveurs modernes sont capables de produire de bonnes solutions avant convergence (ils sont même réglés pour cela prioritairement).

Problème d'optimisation déterministe

Modèle : Programme mathématique

$$\begin{array}{lll} min & & f(x) \\ s.c. & g_k(x) \leq b_k & \forall k \\ & x \in X \subseteq \mathbb{R}^n & \end{array}$$

Différents types d'optimisation

- Pas de g_k, $X = \mathbb{R}^n$: *optimisation sans contraintes*
- f, g_k linéaires : *Programmation Linéaire*
- X discret ($\subseteq \mathbb{Z}^n$) : *Programmation Entière*

Optimisation dans l'incertain

Problématiques courantes

- Planification des maintenances de centrales nucléaires
  demande en électricité, capacité de production, pannes...
- Optimisation de traitements phytosanitaires
  météo, développement de maladies...
- Ordonnancement de processus informatiques
  arrivée de nouveaux processus en temps réel...
- Routage maritime à propulsion mixte météo
- Investissements à moyen/long termes
  prévisions de demande, vente...
- Trafic routier, ferroviaire, aérien, maritime...
- Processus industriels de pointe (microélectronique)
  état courant du système incertain, effet des décisions incertain

Optimisation stochastique

- Objectif : solution bonne en moyenne
- Pré-requis : statistiques précises sur les paramètres
  lois de probabilité, ensemble de scénarios représentatifs. . .

Approche orientée "moyenne"

$$\begin{array}{ll} \min & t \\ s.c. & t \geq \mathbb{E}_{\xi \in \Xi}[Q(x,\xi)] \\ & g(x) \leq 0 \\ & x \in X, t \in \mathbb{R} \end{array}$$

$Q(x,\xi)$: coût de la solution x sous l'aléa ξ (inclut l'irréalisabilité)

Approches "risk-averse" : (Conditional) Value-At-Risk, variance. . .

Optimisation robuste

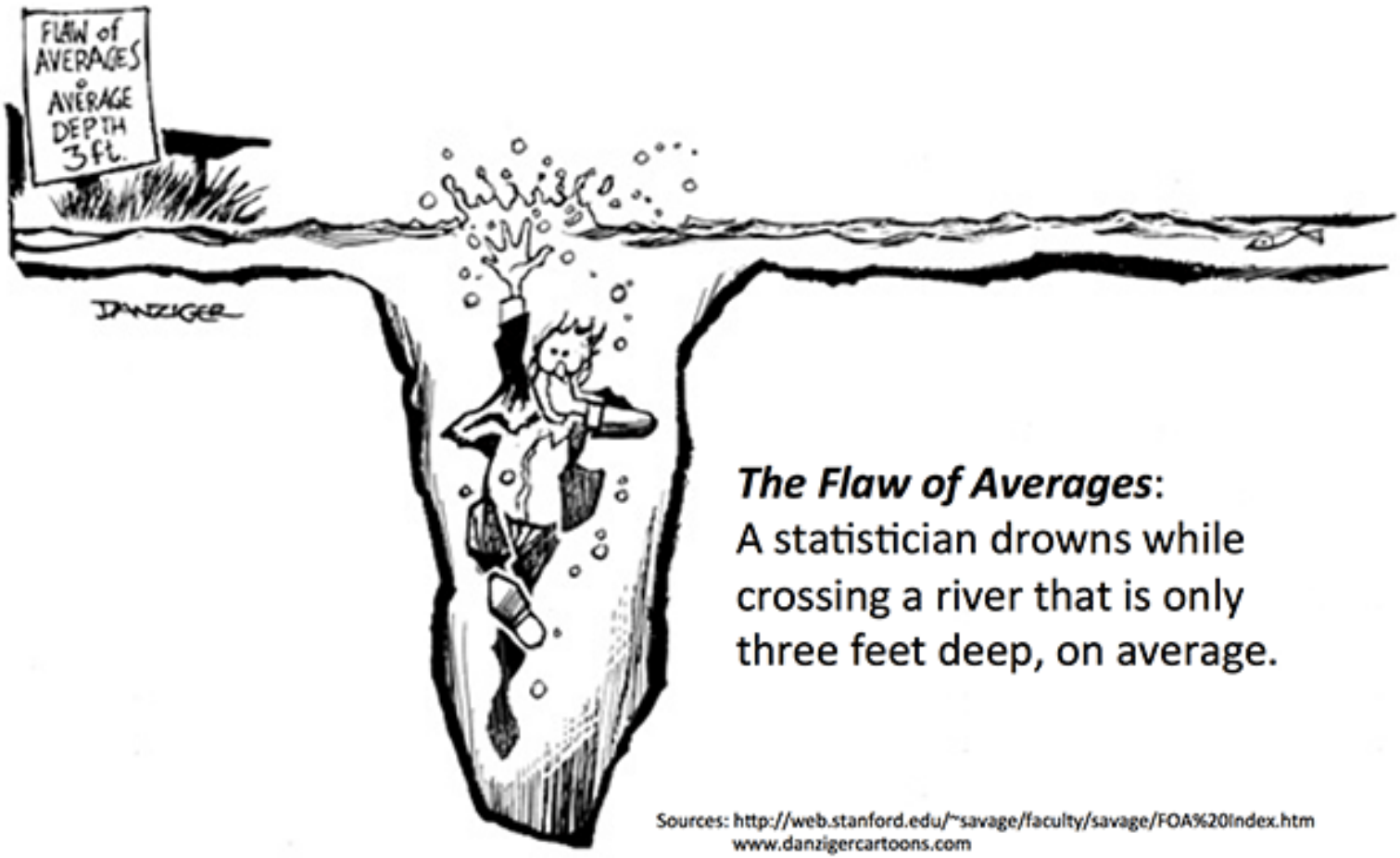
- Objectif : **immuniser** contre les événements incertains les plus néfastes
  - solution réalisable ou réparable dans tous les cas
  - solution bonne dans le pire des cas
- Pré-requis : description des paramètres plausibles Ξ
- Souvent plus facile à traiter en pratique

Approche pire cas

$$\begin{array}{ll} \min & t \\ s.c. & \left.\begin{array}{l} t \geq Q(x,\xi) \\ g(x,\xi) \leq 0 \end{array}\right\} \forall \xi \in \Xi \\ & x \in X, t \in \mathbb{R} \end{array}$$

$Q(x,\xi)$: coût de la solution x sous l'aléa ξ

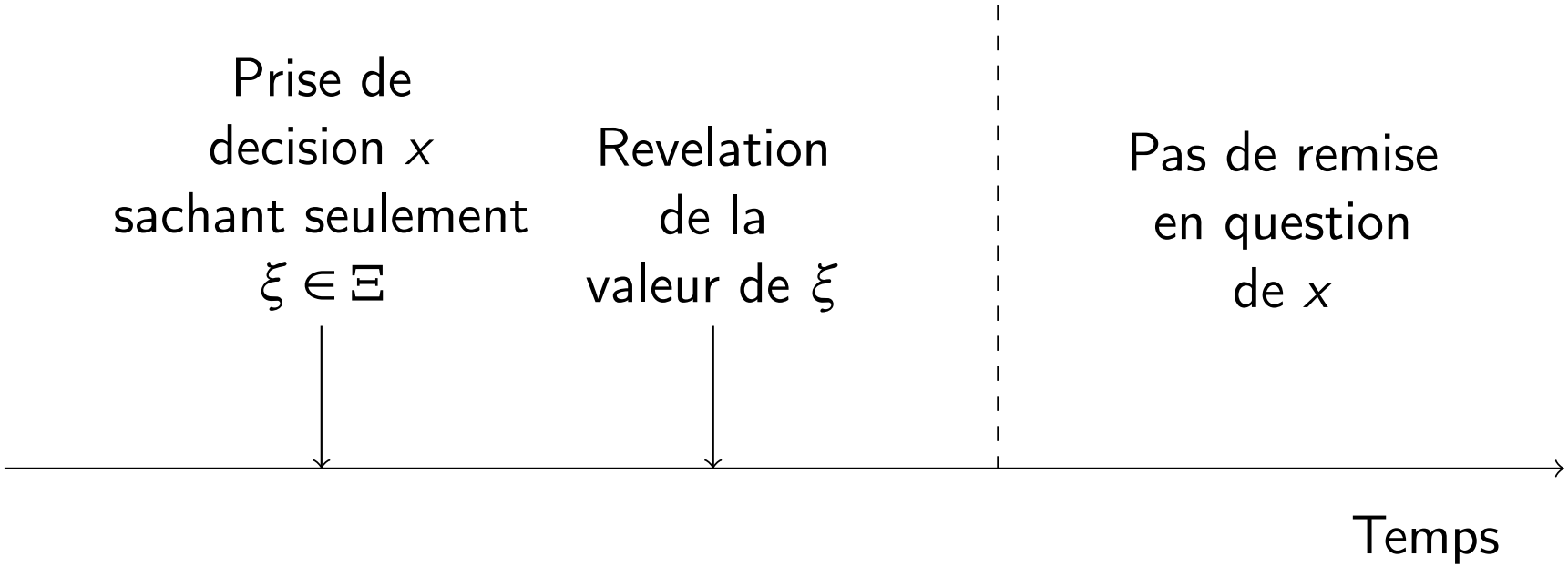
Stochastique VS Robuste



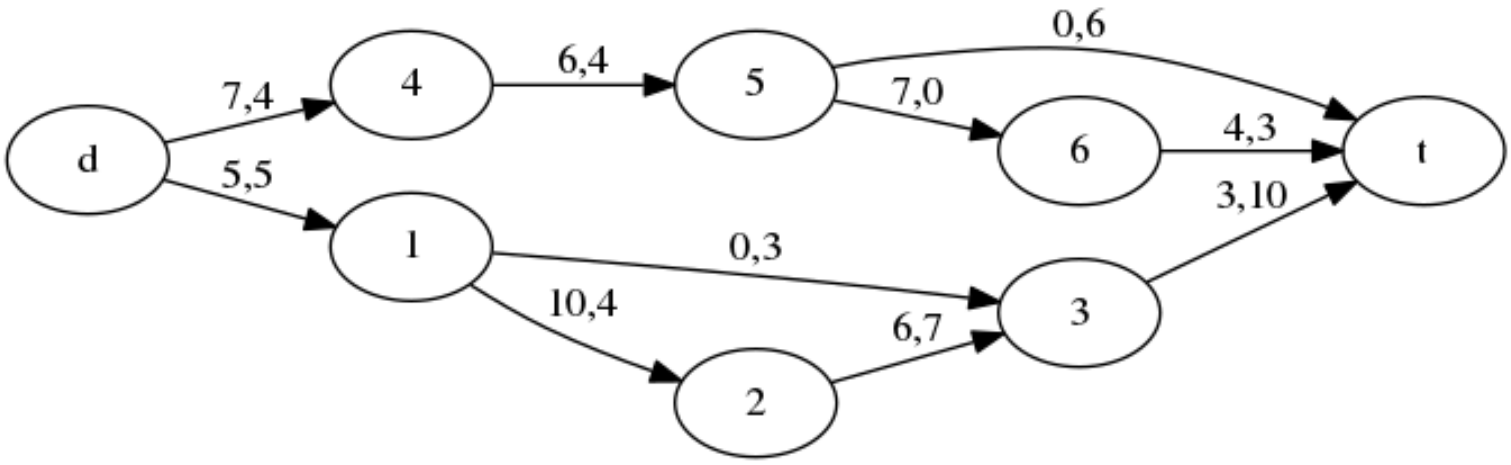
The Flaw of Averages:
A statistician drowns while crossing a river that is only three feet deep, on average.

Sources: http://web.stanford.edu/~savage/faculty/savage/FOA%20Index.htm
www.danzigercartoons.com

Modèles statiques (sans recours)



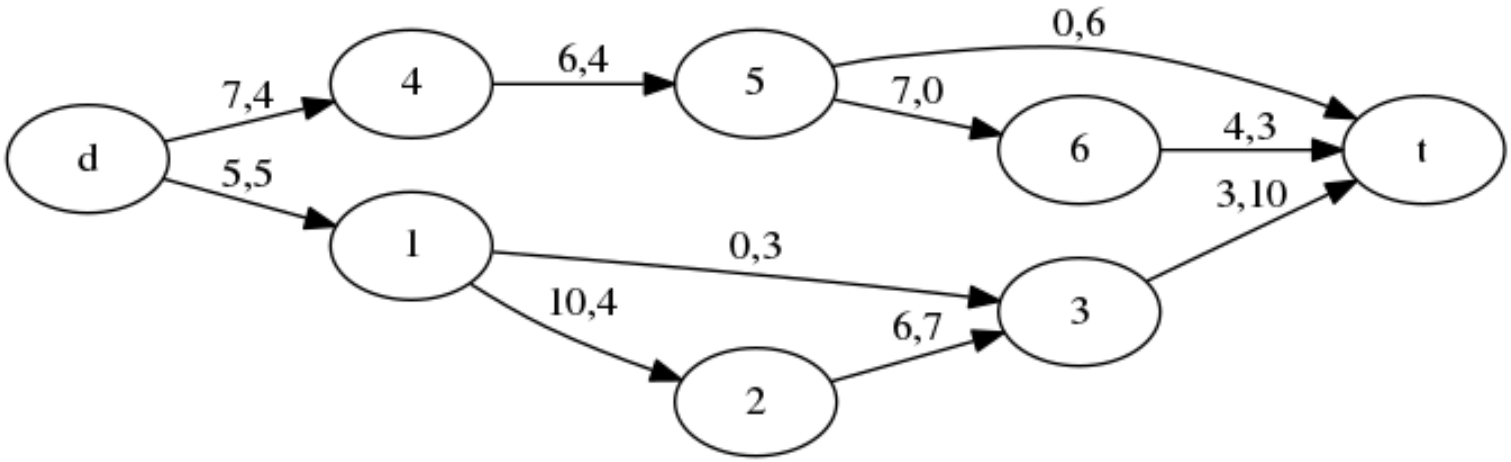
Exemple : plus court chemin avec coûts incertains



Modélisation de l'incertitude

- Ensemble S de scénarios
- Chaque scénario $s \in S$ définit un coût $c^s_{i,j}$ pour chaque arc (i,j)
- Coût d'un chemin μ dans le scénario s : $z(\mu) = \sum_{(i,j)\in\mu} c^s_{i,j}$
- Coût robuste de μ : $z(\mu) = \max_{s\in S} z(\mu)$

Exemple : plus court chemin avec coûts incertains

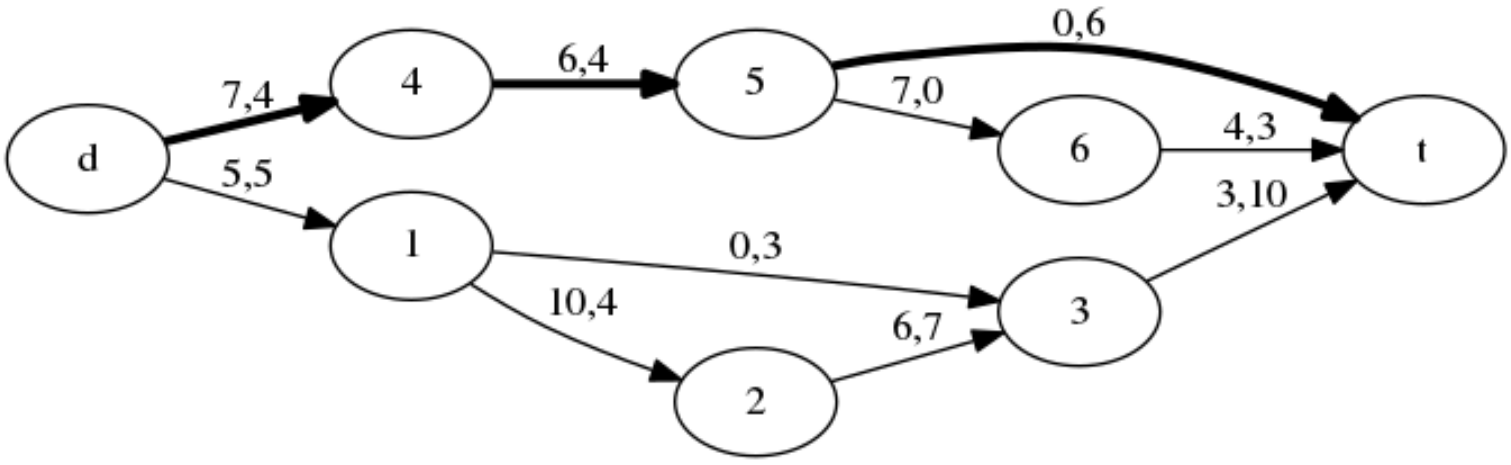


Problème robuste

Trouver un chemin μ^* de d à t tel que $z(\mu^*) = \min_\mu z(\mu)$

NP-difficile même pour deux scénarios et un graphe acyclique [Yu et Yang, 1998]

Exemple : plus court chemin avec coûts incertains



Problème robuste

Trouver un chemin μ^* de d à t tel que $z(\mu^*) = \min_\mu z(\mu)$

NP-difficile même pour deux scénarios et un graphe acyclique [Yu et Yang, 1998]

Exemple : plus court chemin avec coûts incertains

Programme Linéaire en Nombres Entiers

$$
\begin{aligned}
\min \quad & t \\
s.c. \quad & t \geq \sum_{(i,j)\in A} c_{ij}^{s} x_{ij} && \forall s \in S \\
& \sum_{j:(j,i)\in A} x_{ji} - \sum_{j:(i,j)\in A} x_{ij} = \begin{cases} 1 & \text{si } i = d \\ -1 & \text{si } i = t \\ 0 & \text{sinon} \end{cases} && \forall i \in V \\
& x_{ij} \in \{0,1\} && \forall (i,j) \in A
\end{aligned}
$$

Méthode de résolution générique

$$\begin{array}{ll} \min & t \\ s.c. & \left.\begin{array}{l} t \geq Q(x,\xi) \\ g(x,\xi) \leq 0 \end{array}\right\} \forall \xi \in \Xi \\ & x \in X, t \in \mathbb{R} \end{array}$$

Méthode de plans sécants / Branch-and-cut

1. Considérer uniquement les conditions nominales
2. Résoudre le problème courant (souvent NP-difficile)
3. Contrôler si la solution est réalisable pour son pire cas (parfois NP-difficile → (CR) est PSPACE-Complet)
4. Si oui, terminé
   Sinon, ajouter ce scénario au problème et retour en 2

Cas particulier 1. Ensemble d'incertitude polyédral

Hypothèses

- Modélisation PL(NE)
  Problème déterministe

$$(P)\ \min \quad cx$$
$$s.c. \quad Ax \leq b$$
$$x \in X \subseteq \mathbb{R}^n$$

  Contrepartie robuste

$$(CR)\ \min \quad cx$$
$$s.c. \quad A(\xi)x \leq b(\xi) \quad \forall \xi \in \Xi$$
$$x \in X \subseteq \mathbb{R}^n$$

- Ξ borné non vide : $\Xi = \left\{\xi \in \mathbb{R}^L : \xi \geq 0, F\xi \leq g\right\}$
- $A(\xi)$ et $b(\xi)$ fonctions affines de ξ

Résultat [Ben-Tal et al., 2009]

Reformulation en PL(mixte) grâce à la dualité de PL

Cas particuliers 2. Variables binaires et coût incertain

Hypothèses

- Le coût de Γ variables peut dévier : $\tilde{c}_j \in [c_j, c_j + d_j]$
- Modélisation PL en variables binaires

Problème déterministe

$$(P)\ \min \quad cx$$
$$s.c. \quad Ax \leq b$$
$$x \in \{0,1\}^n$$

Contrepartie robuste

$$(CR)\ \min \quad f(x)$$
$$s.c. \quad Ax \leq b$$
$$x \in \{0,1\}^n$$

avec $f(x) = cx + \max_{S:S\subseteq N,|S|\leq\Gamma} \sum_{j\in S} d_j x_j$

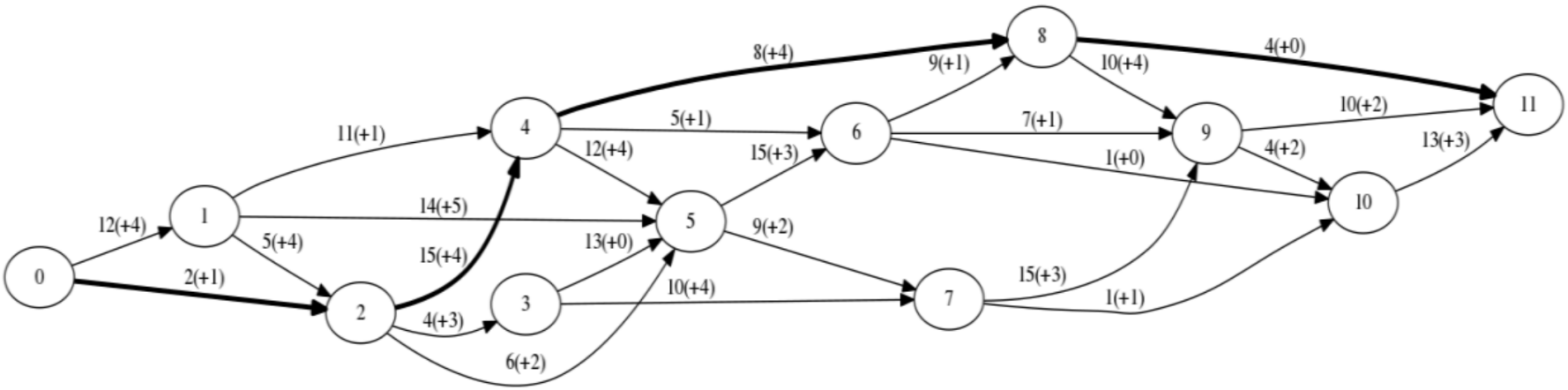
Résultat [Bertsimas et Sim, 2003]

(CR) résolvable comme $n+1$ problèmes $\min\{c'x : Ax \leq b, x \in \{0,1\}^n\}$

Cas particuliers : 2. Variables binaires et coût incertain

Plus court chemin avec coûts en intervalle, au plus Γ déviations

$n+1$ problèmes de plus courts chemins avec des coûts modifiés



Solution optimale : $\mu^* = (0, 2, 4, 8, 11)$, coût dans $[29; 39]$.

Avantages et inconvénients des modèles statiques

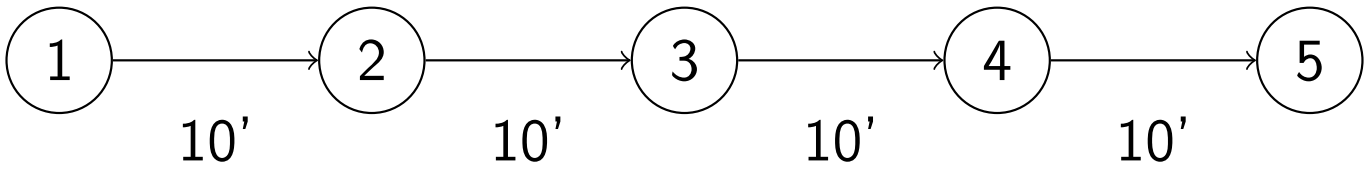
Avantages

Dans de nombreux cas, le problème reste de la même classe de complexité.
Les mêmes algorithmes peuvent parfois être utilisés pour le problème déterministe et sa variante robuste.

Inconvénients

- Comment déterminer l'ensemble d'incertitude ?
  (en général plus facile que déterminer des lois de probabilité)
- Peut être délicat à modéliser.
  Dans les modèles robustes sans recours, l'incertitude, même couplée, s'applique indépendamment sur chaque contrainte.

Exemple : horaires sur une ligne de train



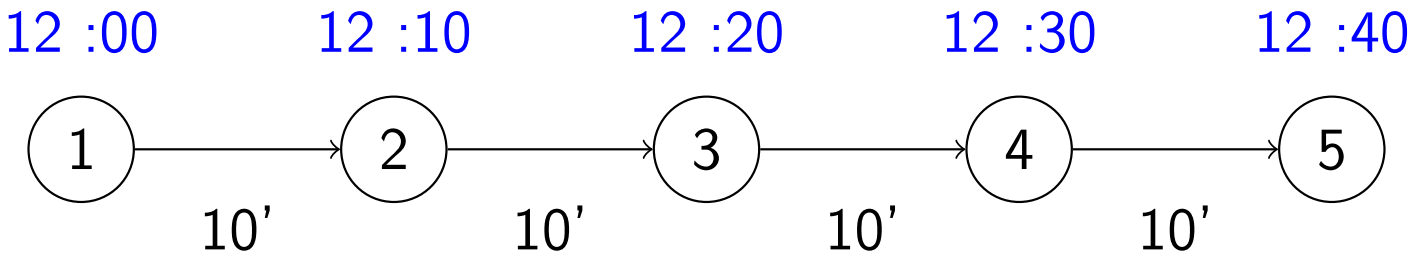
Problème

Déterminer une heure de départ à chaque arrêt

- arriver au plus tôt au dernier arrêt
- pouvoir ne jamais partir en retard ni en avance

$$
\begin{aligned}
\min \quad & t_5 - t_1 \\
s.c. \quad & t_2 - t_1 \geq 10 \\
& t_3 - t_2 \geq 10 \\
& t_4 - t_3 \geq 10 \\
& t_5 - t_4 \geq 10 \\
& t \geq 0
\end{aligned}
$$

Exemple : horaires sur une ligne de train



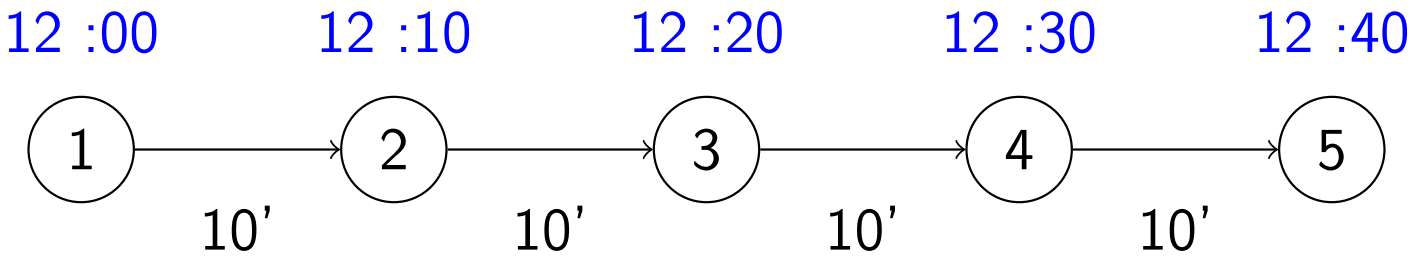
Problème

Déterminer une heure de départ à chaque arrêt

- arriver au plus tôt au dernier arrêt
- pouvoir ne jamais partir en retard ni en avance

$$
\begin{aligned}
\min \quad & t_5 - t_1 \\
s.c. \quad & t_2 - t_1 \geq 10 \\
& t_3 - t_2 \geq 10 \\
& t_4 - t_3 \geq 10 \\
& t_5 - t_4 \geq 10 \\
& t \geq 0
\end{aligned}
$$

Exemple : horaires sur une ligne de train



Problème

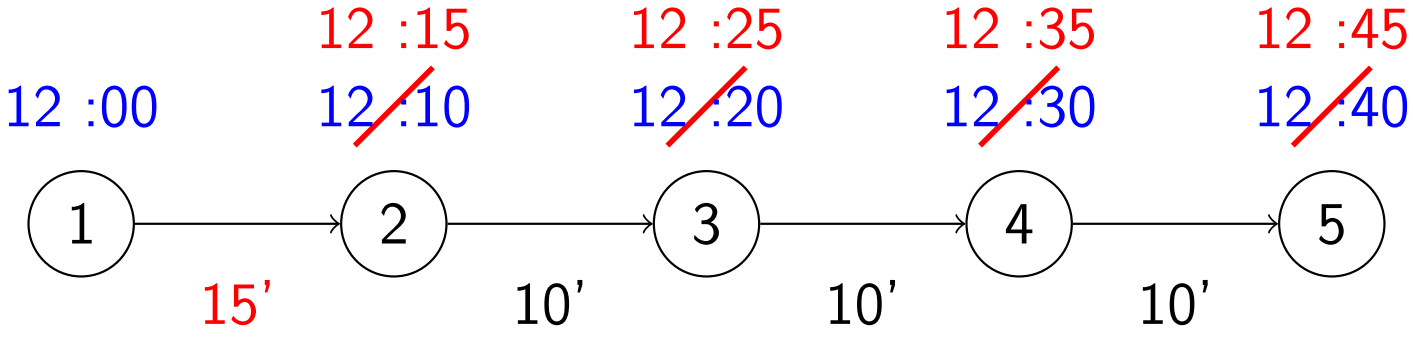
Déterminer une heure de départ à chaque arrêt

- arriver au plus tôt au dernier arrêt
- pouvoir ne jamais partir en retard ni en avance

On veut se protéger contre un délai total de 5'.

$$\left.\begin{array}{ll} \min & t_5 - t_1 \\ s.c. & t_2 - t_1 \geq 10 + \xi_1 \\ & t_3 - t_2 \geq 10 + \xi_2 \\ & t_4 - t_3 \geq 10 + \xi_3 \\ & t_5 - t_4 \geq 10 + \xi_4 \\ & t \geq 0 \end{array}\right\} \quad \begin{array}{l} \forall \xi \geq 0 : \\ \sum_{i=1}^{4} \xi_i \leq 5 \end{array}$$

Exemple : horaires sur une ligne de train



Problème

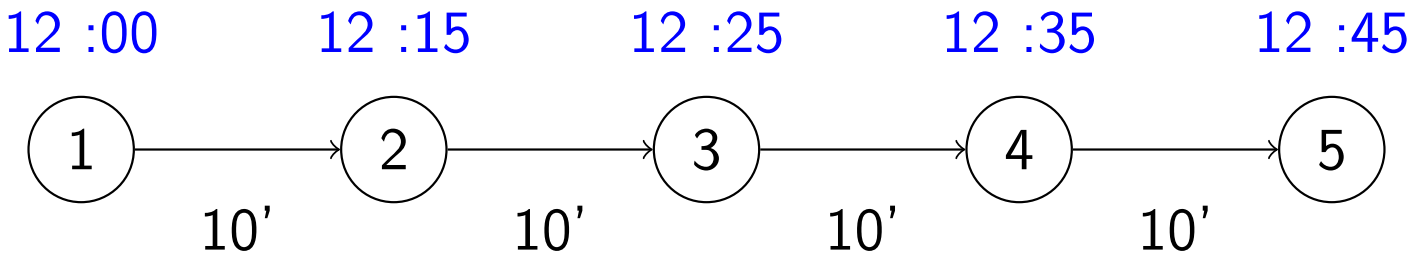
Déterminer une heure de départ à chaque arrêt

- arriver au plus tôt au dernier arrêt
- pouvoir ne jamais partir en retard ni en avance

On veut se protéger contre un délai total de 5'.

$$\left.\begin{array}{ll} \min & t_5 - t_1 \\ s.c. & t_2 - t_1 \geq 10 + \xi_1 \\ & t_3 - t_2 \geq 10 + \xi_2 \\ & t_4 - t_3 \geq 10 + \xi_3 \\ & t_5 - t_4 \geq 10 + \xi_4 \\ & t \geq 0 \end{array}\right\} \quad \begin{array}{l} \forall \xi \geq 0 : \\ \sum_{i=1}^{4} \xi_i \leq 5 \end{array}$$

Exemple : horaires sur une ligne de train



Problème

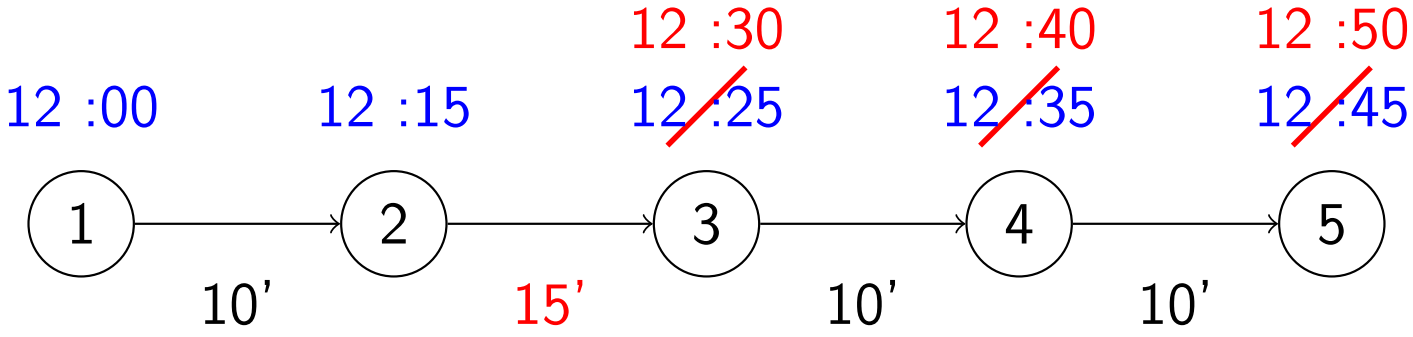
Déterminer une heure de départ à chaque arrêt

- arriver au plus tôt au dernier arrêt
- pouvoir ne jamais partir en retard ni en avance

On veut se protéger contre un délai total de 5'.

$$\left.\begin{array}{ll} \min & t_5 - t_1 \\ s.c. & t_2 - t_1 \geq 10 + \xi_1 \\ & t_3 - t_2 \geq 10 + \xi_2 \\ & t_4 - t_3 \geq 10 + \xi_3 \\ & t_5 - t_4 \geq 10 + \xi_4 \\ & t \geq 0 \end{array}\right\} \quad \begin{array}{l} \forall \xi \geq 0 : \\ \sum_{i=1}^{4} \xi_i \leq 5 \end{array}$$

Exemple : horaires sur une ligne de train



Problème

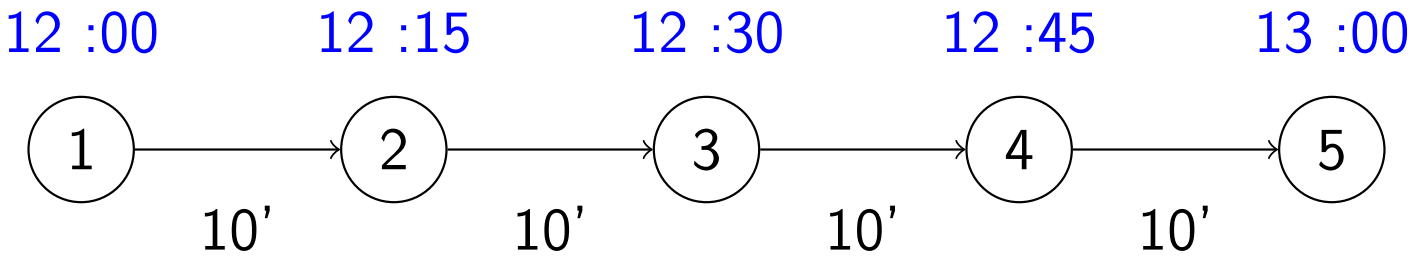
Déterminer une heure de départ à chaque arrêt

- arriver au plus tôt au dernier arrêt
- pouvoir ne jamais partir en retard ni en avance

On veut se protéger contre un délai total de 5'.

$$\left.\begin{array}{ll} \min & t_5 - t_1 \\ s.c. & t_2 - t_1 \geq 10 + \xi_1 \\ & t_3 - t_2 \geq 10 + \xi_2 \\ & t_4 - t_3 \geq 10 + \xi_3 \\ & t_5 - t_4 \geq 10 + \xi_4 \\ & t \geq 0 \end{array}\right\} \quad \begin{array}{l} \forall \xi \geq 0 : \\ \sum_{i=1}^{4} \xi_i \leq 5 \end{array}$$

Exemple : horaires sur une ligne de train



Problème

Déterminer une heure de départ à chaque arrêt

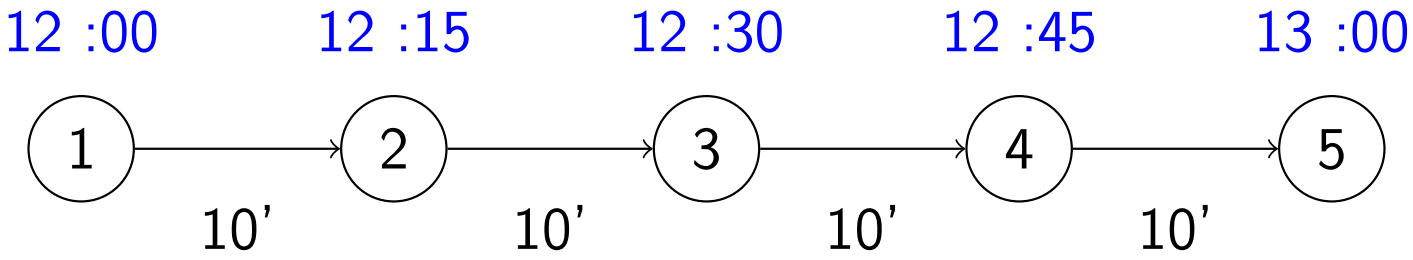
- arriver au plus tôt au dernier arrêt
- pouvoir ne jamais partir en retard ni en avance

On veut se protéger contre un délai total de 5'.

$$\left.\begin{array}{ll} \min & t_5 - t_1 \\ s.c. & t_2 - t_1 \geq 10 + \xi_1 \\ & t_3 - t_2 \geq 10 + \xi_2 \\ & t_4 - t_3 \geq 10 + \xi_3 \\ & t_5 - t_4 \geq 10 + \xi_4 \\ & t \geq 0 \end{array}\right\} \quad \begin{array}{l} \forall \xi \geq 0 : \\ \sum_{i=1}^{4} \xi_i \leq 5 \end{array}$$

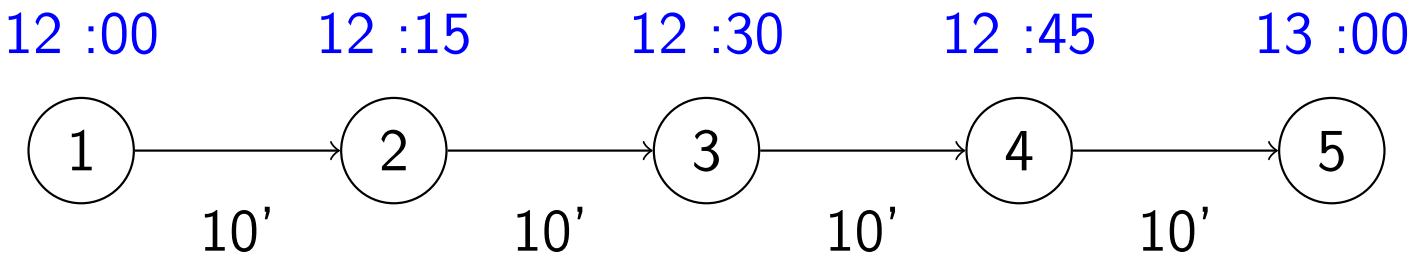
$\Rightarrow$ Décalage au dernier arrêt de 20 minutes

Exemple : horaires sur une ligne de train



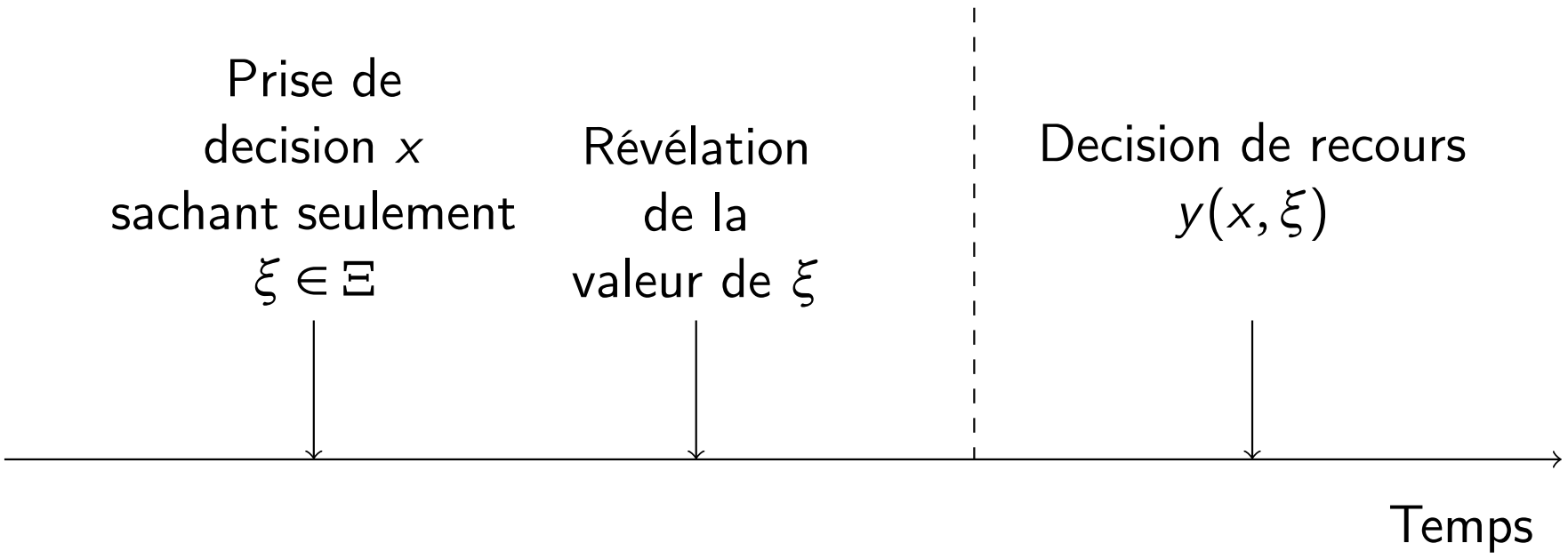
$$
\begin{aligned}
\min \quad & t_5 - t_1 \\
s.c. \quad & t_2 - t_1 \geq 10 + \xi_1 \qquad \forall \xi \geq 0 : \sum_{i=1}^{4} \xi_i \leq 5 \\
& t_3 - t_2 \geq 10 + \xi_2 \qquad \forall \xi \geq 0 : \sum_{i=1}^{4} \xi_i \leq 5 \\
& t_4 - t_3 \geq 10 + \xi_3 \qquad \forall \xi \geq 0 : \sum_{i=1}^{4} \xi_i \leq 5 \\
& t_5 - t_4 \geq 10 + \xi_4 \qquad \forall \xi \geq 0 : \sum_{i=1}^{4} \xi_i \leq 5 \\
& t \geq 0
\end{aligned}
$$

Exemple : horaires sur une ligne de train



$$
\begin{aligned}
\min \quad & t_5 - t_1 \\
s.c. \quad & t_2 - t_1 \geq 15 \\
& t_3 - t_2 \geq 15 \\
& t_4 - t_3 \geq 15 \\
& t_5 - t_4 \geq 15 \\
& t \geq 0
\end{aligned}
$$

Modèles à deux niveaux (avec recours)



Définition formelle

Modèle PL(NE)

$$
\begin{aligned}
(CR): \min \quad & cx + t & \\
& Ax \leq b & \\
& x \in X & \\
& t \geq q(\xi)y(\xi) & \forall \xi \\
& T(\xi)x + W(\xi)y(\xi) \leq h(\xi) & \forall \xi \\
& y(\xi) \in Y & \forall \xi
\end{aligned}
$$

- x : décisions de planification
- t : coût de recours
- $y(\xi)$: décision de recours
- $q(\xi), T(\xi), W(\xi), h(\xi)$: aléa révélé après la planification

Complexité

PSPACE-Complet, mais certains cas restent faciles.

Méthode de résolution

$$
\begin{aligned}
(CR) : min \quad & cx + t \\
& Ax \leq b \\
& x \in X \\
t \geq q(\xi)y(\xi) \quad & \forall \xi \in \Xi \\
T(\xi)x + W(\xi)y(\xi) \leq h(\xi) \quad & \forall \xi \in \Xi \\
y(\xi) \in Y \quad & \forall \xi \in \Xi
\end{aligned}
$$

Méthode de plans sécants / Branch-and-cut

1. Considérer uniquement les conditions nominales
2. Résoudre le problème courant (souvent NP-difficile)
3. Contrôler si la solution est réalisable pour son pire cas
   (Parfois PSPACE-Complet : problème d'optimisation robuste)
4. Si oui, terminé
   Sinon, ajouter ce scénario au problème et retour en 2

Cas particulier : recours affine

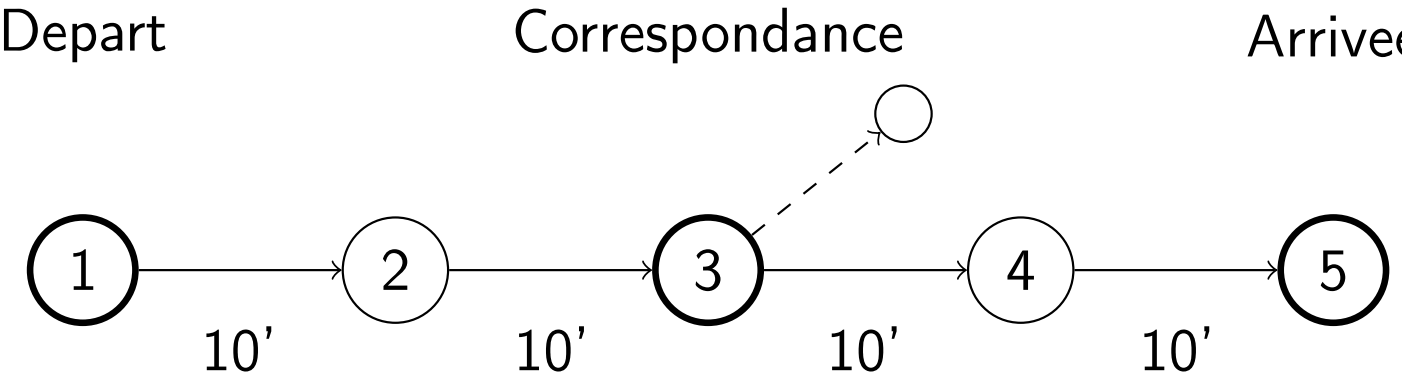
$$
\begin{aligned}
(CR) : \min \quad & cx + t \\
& Ax \leq b \\
& x \in X \\
& t \geq q(\xi)y(\xi) && \forall \xi \in \Xi \\
& T(\xi)x + W(\xi)y(\xi) \leq h(\xi) && \forall \xi \in \Xi \\
& y(\xi) \in Y && \forall \xi \in \Xi
\end{aligned}
$$

Hypothèses

- Recours fixe : W est fixe.
- $T(\xi)$ et $h(\xi)$ sont des fonctions affines de ξ
- Ξ est polyédral, borné non vide : $\Xi = \{\xi \in \mathbb{R}^L : \xi \geq 0, F\xi \leq g\}$
- $y(\xi)$ est une fonction affine de ξ

Résultat [Ben-Tal et al., 2004]

Exemple : horaires sur une ligne de train



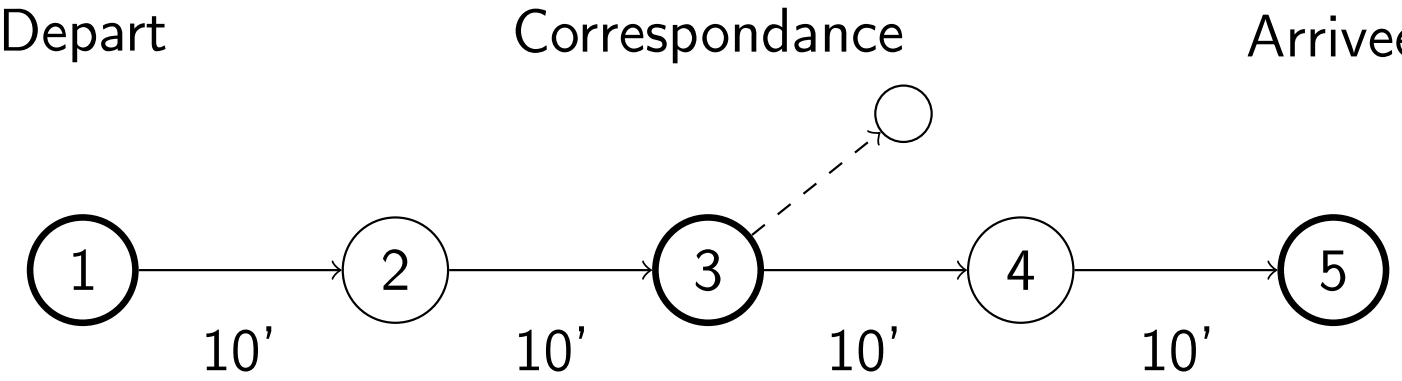
Problème

- Fixer une heure de départ sauf en gare desserte
- Ajuster les heures de départ en desserte
- arriver au plus tôt au dernier arrêt

$$\left.\begin{array}{ll} \min & t_5 - t_1 \\ s.c. & t_2(\xi) - t_1 \geq 10 + \xi_1 \\ & t_3 - t_2(\xi) \geq 10 + \xi_2 \\ & t_4(\xi) - t_3 \geq 10 + \xi_3 \\ & t_5 - t_4(\xi) \geq 10 + \xi_4 \\ & t \geq 0 \end{array}\right\} \quad \forall \xi \geq 0 : \sum_{i=1}^{4} \xi_i \leq 5$$

Protection délai total de 5'.

Exemple : horaires sur une ligne de train



Problème

- Fixer une heure de départ sauf en gare desserte
- Ajuster les heures de départ en desserte
- arriver au plus tôt au dernier arrêt

$$\left.\begin{array}{ll} \min & t_5 - t_1 \\ s.c. & t_2(\xi) - t_1 \geq 10 + \xi_1 \\ & t_3 - t_2(\xi) \geq 10 + \xi_2 \\ & t_4(\xi) - t_3 \geq 10 + \xi_3 \\ & t_5 - t_4(\xi) \geq 10 + \xi_4 \\ & t \geq 0 \end{array}\right\} \quad \forall \xi \geq 0 : \sum_{i=1}^{4} \xi_i \leq 5$$

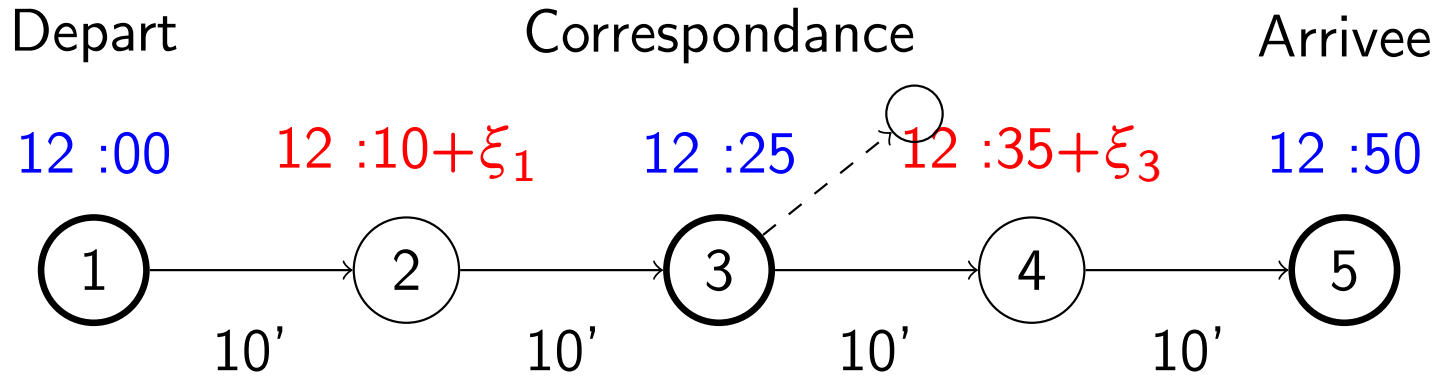
$$t_2(\xi) = t_2^0 + t_2^1 \xi_1$$

$$t_4(\xi) = t_4^0 + t_4^1 \xi_1 + t_4^2 \xi_2 + t_4^3 \xi_3$$

Protection délai total de 5'.

$\Rightarrow$ Modèle robuste statique

Exemple : horaires sur une ligne de train



Problème

- Fixer une heure de départ sauf en gare desserte
- Ajuster les heures de départ en desserte
- arriver au plus tôt au dernier arrêt

Protection délai total de 5'.

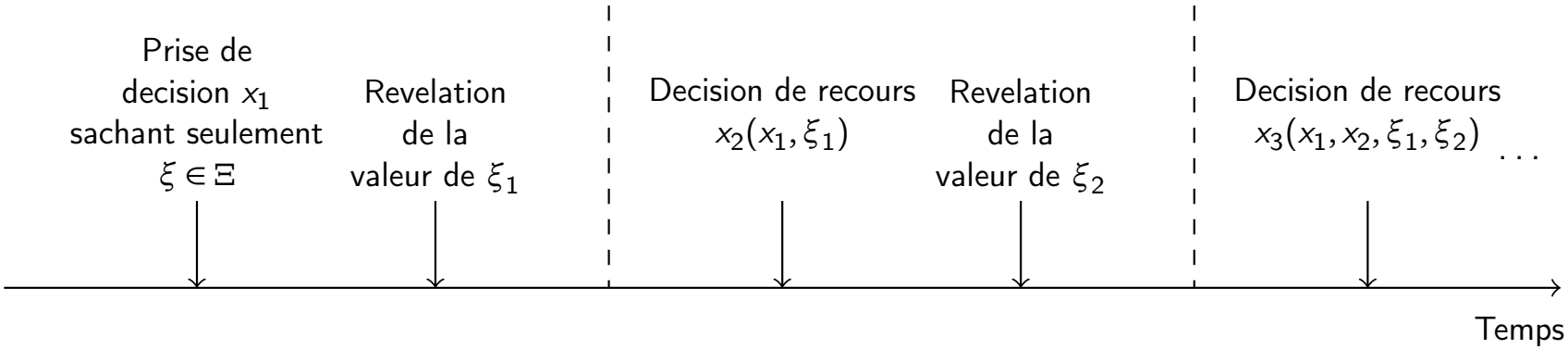
$$\left.\begin{array}{ll} \min & t_5 - t_1 \\ s.c. & t_2(\xi) - t_1 \geq 10 + \xi_1 \\ & t_3 - t_2(\xi) \geq 10 + \xi_2 \\ & t_4(\xi) - t_3 \geq 10 + \xi_3 \\ & t_5 - t_4(\xi) \geq 10 + \xi_4 \\ & t \geq 0 \end{array}\right\} \quad \begin{array}{l} \forall \xi \geq 0 : \\ \sum_{i=1}^{4} \xi_i \leq 5 \end{array}$$

$$t_2(\xi) = t_2^0 + t_2^1 \xi_1$$

$$t_4(\xi) = t_4^0 + t_4^1 \xi_1 + t_4^2 \xi_2 + t_4^3 \xi_3$$

$\Rightarrow$ Modèle robuste statique

Modèles multi-niveaux



Problèmes typiques

- Planification sur un horizon temporel avec adaptation des décisions
- Jeu à deux joueurs
- ...

Une solution est une politique : étant donné l'état courant du système, quelle est la meilleure décision ?

Optimisation robuste multi-niveaux

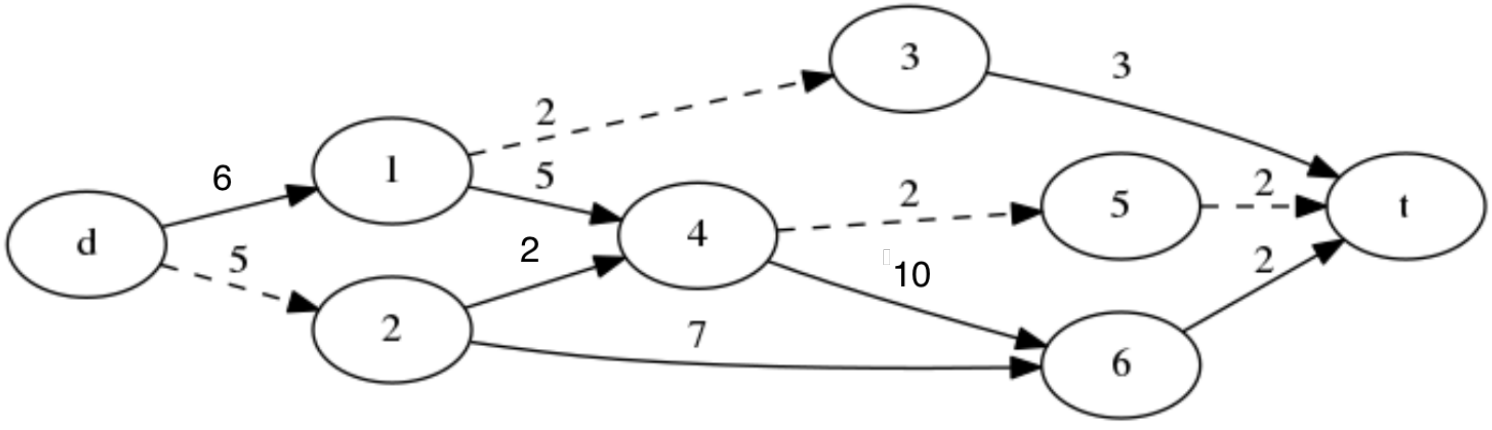
Niveau	Informations disponibles			Décision
	Décisions antérieures	Aléa observé	Incertitude résiduelle	
1	-	-	$(\xi_1, \ldots, \xi_K)$	x_1
2	x_1	ξ_1	$(\xi_2, \ldots, \xi_K \vert \xi_1)$	x_2
3	x_1, x_2	ξ_1, ξ_2	$(\xi_3, \ldots, \xi_K \vert \xi_1, \xi_2)$	x_3
...				
K	$x_1, \ldots, x_{K-1}$	$\xi_1, \ldots, \xi_{K-1}$	$(\xi_K \vert \xi_1, \ldots, \xi_{K-1})$	x_K
K+1	$x_1, \ldots, x_K$	$\xi_1, \ldots, \xi_K$	-	x_{K+1}

Optimisation robuste multi-niveaux

$$
\begin{aligned}
\min z = c^1x^1 + \max_{\xi^1}[\min c^2(\omega)x^2(\omega^2)+ \quad & \cdots \\
+ \max_{\xi^K}[\quad \min \quad & c^K(\omega)x^K(\omega^K)]\cdots] \\
s.t. \qquad W^1x^1 \quad &= \quad h^1 \\
T^1(\omega)x^1 + W^2x^2(\omega^2) \quad &= \quad h^2(\omega) \\
\cdots \quad &\vdots \\
T^{K-1}(\omega)x^{K-1}(\omega^{K-1}) + W^Kx^K(\omega^K) \quad &= \quad h^K(\omega) \\
T^K(\omega)x^K + W^{K+1}x^{K+1}(\omega^K) \quad &= \quad h^{K+1}(\omega) \\
x^1 \geq 0, x^K(\omega^K) \geq 0 \quad & \qquad k = 2,\ldots,K+1
\end{aligned}
$$

- ω : scénario défini sur $\{1,\ldots,K\}$
- ω^k : scénario restreint à $\{1,\ldots,k\}$

Exemple : plus court chemin robuste avec pannes



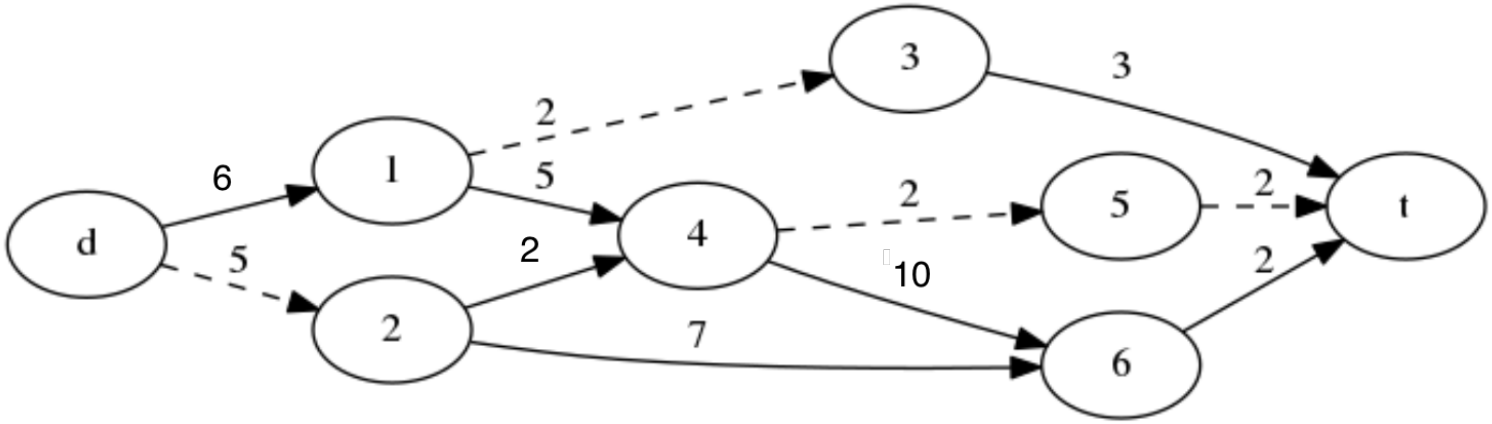
Données

- Graghe $G = (V, A, c)$
- Arcs fragiles $F \subseteq A$; Γ arcs de F peuvent casser
- On découvre si un arc est cassé sur son sommet initial

Objectif

Déterminer une politique qui donnera, dans le pire des cas, le chemin le plus court : sachant γ pannes restantes, quel prochain sommet choisir ?

Exemple : plus court chemin robuste avec pannes



Résolution par prog. dynamique (simpl. pour exposé)

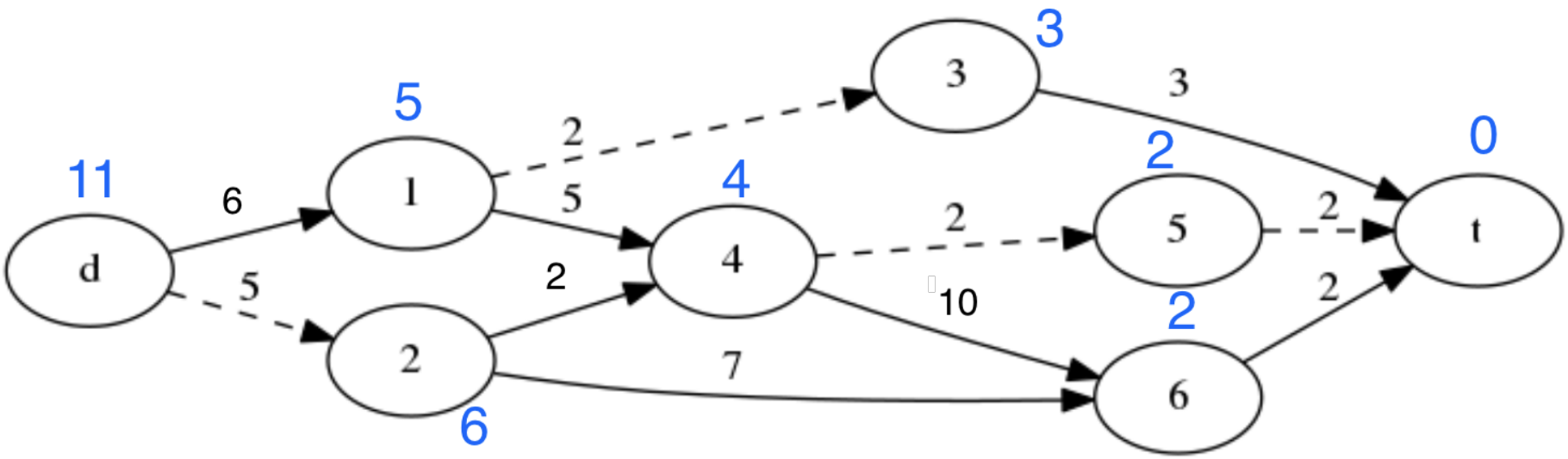
Coût optimal de u à t avec au plus l pannes :

$$d_u^l = \begin{cases} \min\{d_{uv}^l | (u,v) \in E\} & \text{if } u \neq t \\ 0 & \text{if } u = t \end{cases}$$

Coût optimal de u à t avec au plus l pannes en planifiant (u,v) :

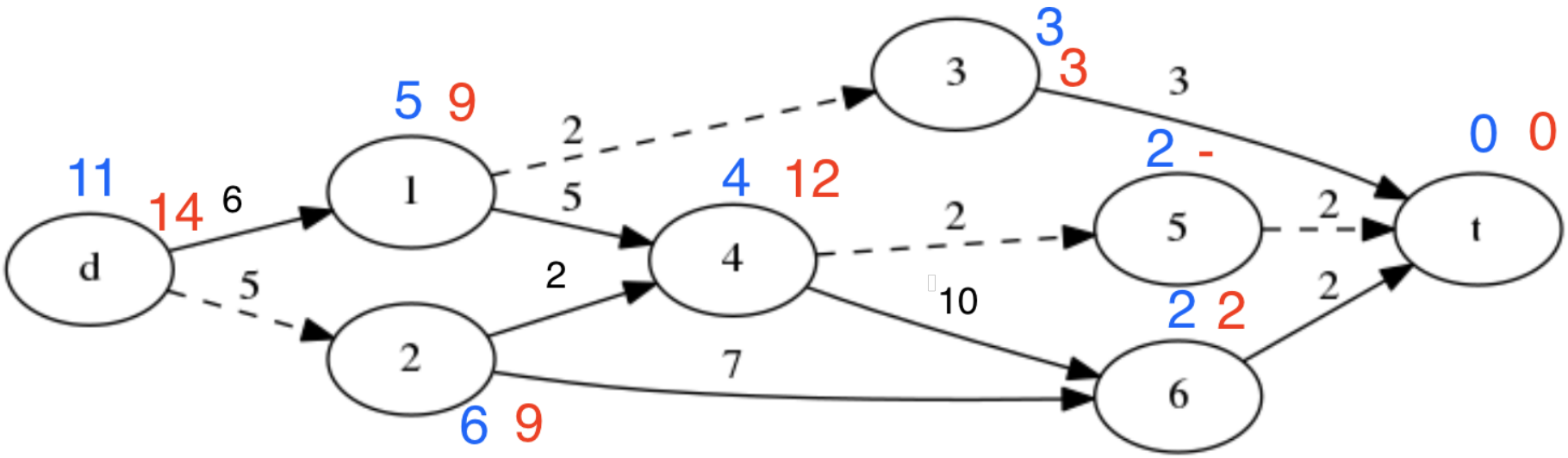
$$d_{uv}^l = \begin{cases} d_v^l + c_{uv} & \text{if } (u,v) \notin F \\ \max\{d_v^l + c_{uv}, \min\{d_w^{l-1} + c_{uw} | (u,w) \in E, w \neq v\}\} & \text{if } (u,v) \in F \end{cases}$$

Exemple : plus court chemin robuste avec pannes



Distances au puits sans panne

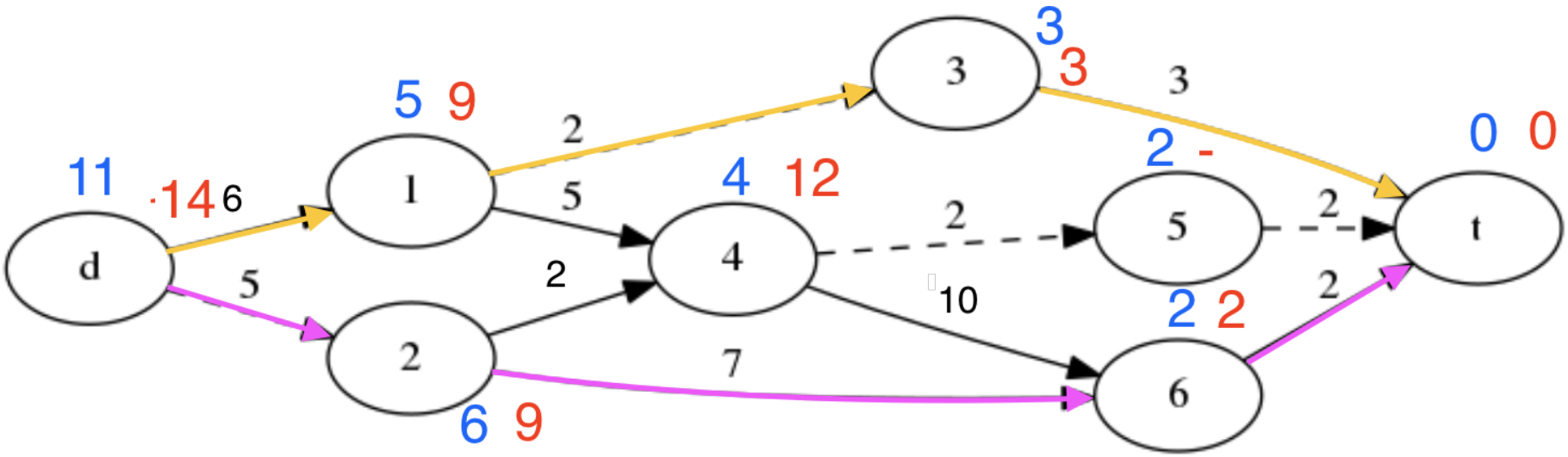
Exemple : plus court chemin robuste avec pannes



Distances au puits sans panne

Distances au puits avec une panne (avec recours)

Exemple : plus court chemin robuste avec pannes



Distances au puits sans panne

Distances au puits avec une panne (avec recours)

Fuschia : Chemin de planification

Orange : Chemin de recours

Conclusion sur les approches robustes

- Grande diversité d'approches proposées
- Gradient de complexité théorique de très simple à très complexe
- Plus délicat à modéliser et mettre en oeuvre que les approches déterministes
- Approche pragmatique : Light Robustness (Fischetti et al., 2009)
  - Modéliser comme un problème robuste statique
  - Rendre les contraintes robustes "souples" en pénalisant leur violation
  - Plusieurs variantes pour déterminer des pénalisations adéquates

Conclusion générale

Les problèmes d'optimisation dans les graphes ont des applications dans de nombreux champs disciplinaires.

Les problèmes déterministes sont de mieux en mieux traités (approches ad-hoc, méthodes de programmation linéaire en nombres entiers)

Les méthodes prenant en compte l'incertitude restent beaucoup plus limitées en terme de taille de problèmes résolus

L'approche robuste semble une alternative de choix pour obtenir des problèmes qu'on peut espérer résoudre en pratique

Défis : modélisation, taille des modèles, garanties en cas de non convergence, ...