Outils pour la logistique

Cours 1 : Introduction & Contexte

ECE – 3EME ANNEE
FILIERE *TRANSPORTS ET MOBILITE*

Cédric BENTZ (CNAM)

Logistique

- Gestion des flux physiques destinée à mettre à disposition d'entités demandeuses (clients) des ressources/produits.

- Nombreux aspects :
  - Aspects organisationnels, intervenant à diverses échelles (stratégique, opérationnelle, etc.),
  - **Aspects quantitatifs/numériques.**

Nombreux aspects potentiels concernés par la logistique

- En amont du processus de production : approvisionnement en matières premières (achat aux fournisseurs, extraction, etc.), etc.
- Durant le processus de production (optimisation de la chaîne de production/du montage).
- En aval du processus de production : stockage et transport de marchandises (frêt), etc.

Notions non étudiées

- Gestion détaillée des commandes fournisseurs,
- Contrôle qualité,
- Emballage,
- Traçabilité, respect de la chaîne du froid,
- Distribution vers le client final, tournées et plannings de livraison,
- Maintenance/SAV,
- Etc.

Exemples de notions étudiées

- Achat de matières premières,
- Ordonnancement de processus/de projets,
- Affectation optimale de ressources, planification de tâches,
- Stockage optimal,
- Transport de marchandises (frêt) et calcul d'itinéraires optimaux,
- Etc.

Contexte de l'UE : modèles

- Objet d'étude = **Méthodes quantitatives pour la logistique (et le transport)** :
  - Nécessité de manipuler des concepts formels (mathématiques / informatiques).
  - Méthodes issues de l'optimisation mathématique et de la recherche opérationnelle.
  - Deux principaux types de modèles de base :

(1) graphes,

(2) programmation mathématique.

Contexte de l'UE : méthodologie

- Deux principaux axes d'étude, formant un tout cohérent et complet :
  - Modélisation de différentes problématiques en lien avec la logistique/le transport via les deux types de modèles formels considérés (graphes et programmation mathématique).
  - Utilisation d'algorithmes adéquats et efficaces (ou de logiciels de résolution -dits **solveurs**- les implémentant) pour résoudre les problèmes formels ainsi obtenus.

« Bons » versus « mauvais » algorithmes : quels enjeux ?

- Un algorithme est une méthode automatisée permettant à un système de calcul (PC, smartphone, etc.) de résoudre un problème donné.
- Ainsi, tout système de navigation par GPS possède (implémente) un algorithme lui permettant de calculer les meilleurs itinéraires.
- Mesure de la qualité/efficacité d'un algorithme :
  - Place mémoire occupée (RAM),
  - Temps de réponse (quantité de calculs effectués).

« Bons » versus « mauvais » algorithmes : un exemple (1/3)

SOLDES
BONS PLANS
VENTES FLASH
OFFRES ADHÉRENTS
OFFRES REMBOURSEMENT
PACKS MICRO

TOUTES NOS OFFRES

TABLETTE TACTILE

TOUTES LES TABLETTES
KOBO BY
MICROSOFT SURFACE
IPAD
SAMSUNG GALAXY
TABLETTE ACER
TABLETTE ASUS, NEXUS
ACCESSOIRE TABLETTE

ORDINATEUR

ORDINATEUR PORTABLE

Espace

Résultats : **1** - **10** sur un total de **42**

1 - 2 - 3 - 4 - 5 › »

	Modèle	Ecran	Processeur	Système d'exploitation	Note :	Expédié	Prix
BON PLAN 	Tablette Microsoft Surface Pro 3 12" 256 Go - Intel Core i5	12 "	Intel Core i5*	Windows 8 Pro**	4,5/5	En Stock En stock	BON PLAN 1229,90€ ~~1299,90€~~ » Ajouter au panier » 6 neufs à partir de 1229,90€
BON PLAN 	Tablette Microsoft Surface Pro 3 12" 128 Go	12 "	Intel Core i5*	Windows 8 Pro**	4,5/5	En Stock En stock	BON PLAN 929,90€ ~~999,90€~~ » Ajouter au panier » 6 neufs à partir de 929,90€
SOLDE 	Pack Tablette Asus T100TA-DK090H 10.1" Tactile, 1 To + Housse + Microsoft Office 2013	10,1 "	Intel Quad-Core Atom Bay Trail-T	Windows 8 32 Bits**	4/5	En Stock	SOLDE 399,98€ ~~499,90€~~ » Ajouter au panier
BON PLAN 	Tablette Microsoft Surface Pro 3 12" 64 Go	12 "	Intel Core i3*	Windows 8 Pro**	3,5/5	En Stock En stock	BON PLAN 749,90€ ~~799,90€~~ » Ajouter au panier » 5 neufs à partir de 749,90€

« Bons » versus « mauvais » algorithmes : un exemple (2/3)

- Trier n entiers par ordre croissant ==> méthode ?
- Opération = comparaison, test, affectation, etc.
- Tri par sélection : pour tout i de 1 à n, trouver le $i^{ème}$ plus petit élément, et le placer en $i^{ème}$ position.
  - Nombre d'opérations dans le pire cas = $O(n^2)$
- Tri plus “rapides” ? Tri par tas ou tri fusion :
  - Nombre d'opérations dans le pire cas = $O(n \log n)$

« Bons » versus « mauvais » algorithmes : un exemple (3/3)

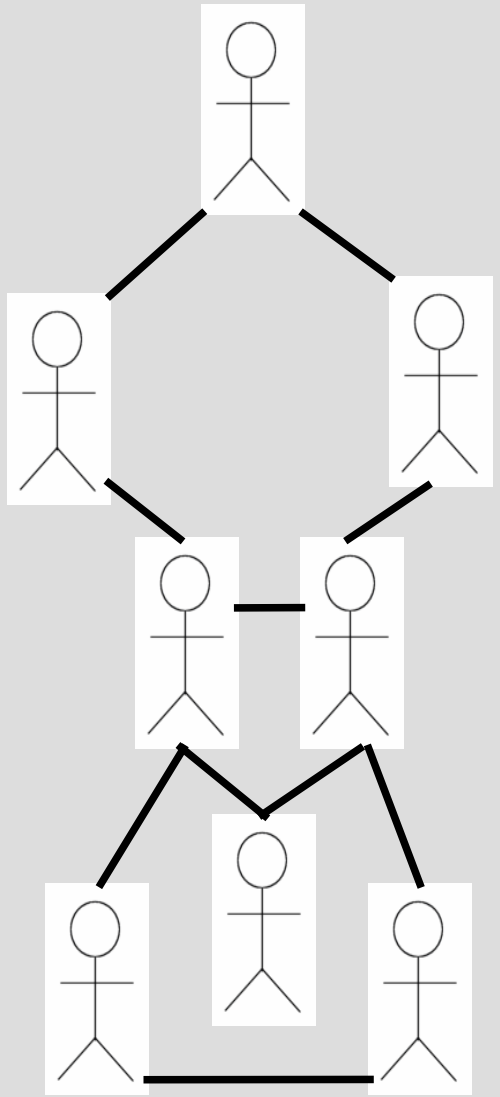
- Sur un PC qui effectue 1 milliard d'opérations par seconde (fréquence CPU = 1 GHz) :
  - Si n = 1 000 000 (1 à 2 Mo) :
    - $n^2 = 10^{12}$ ==> Temps = 1000 secondes
    - n*log n = 6 000 000 ==> Temps = 6 ms
  - Si n = 1 milliard (1 à 2 Go) :
    - $n^2 = 10^{18}$ ==> Temps = 1 milliard de sec. (< 32 ans)
    - n*log n = 9 milliards ==> Temps = 9 secondes

Graphe (rappels)

- Un graphe est un objet formel donné par :
  - Un ensemble de points (**sommets** ou **noeuds**),
  - Un ensemble de traits qui interconnectent ces points (**arcs** si orientés, **arêtes** sinon).
- Quelques exemples ?
  - Réseau ferroviaire (métro parisien ==> arcs),
  - Carte routière,
  - Réseau de distribution (gaz, eau, électricité),
  - Réseau social, web, etc.

Une application des graphes : les réseaux sociaux (RS)

- A chaque utilisateur/membre est associé un sommet/nœud du graphe représentant le RS
- Une arête existe entre 2 membres du RS qui se connaissent/sont amis/sont en relation
- Les grands RS les plus connus (FB, etc.) peuvent alors avoir à gérer des graphes ayant potentiellement des millions ou des milliards de sommets et des centaines de milliards d'arêtes !



Résolution de problèmes via l'optimisation dans les graphes

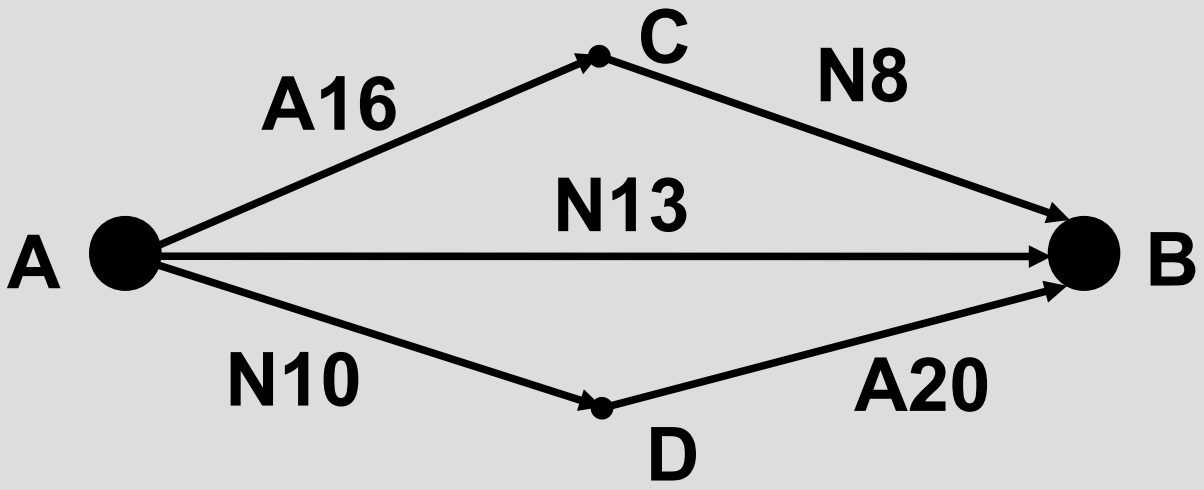
- De nombreuses quantités peuvent être calculées dans un graphe (degré maximum, diamètre, etc.).
- Certaines peuvent être reliées à des grandeurs « concrètes » dans un graphe idoine.
- Intérêt de la démarche ?
  - Modéliser un problème concret comme le calcul d'une certaine quantité dans un graphe associé,
  - Résoudre ensuite le problème formel obtenu via un algo. dédié, et en déduire une solution concrète.

Recherche d'un itinéraire optimal

- Comment aller le plus rapidement possible d'un point A à un point B ?

Plus court chemin dans un graphe (1/2)

- Un automobiliste désire se rendre d'une ville A à une ville B le plus rapidement possible
  - 3 trajets possibles :
    - Prendre l'Autoroute A16, puis la Nationale 8 (N8) à C
    - Prendre la Nationale 13 (N13)
    - Prendre la Nationale 10 (N10), puis l'Autoroute A20 à D



Plus court chemin dans un graphe (2/2)

- Son GPS indique les temps de parcours suivants :
  - A16 : 10 minutes le matin, 15 minutes le soir
  - N8 : 7 minutes le matin, 5 minutes le soir
  - N13 : 18 minutes le matin, 19 minutes le soir
  - N10 : 12 minutes le matin, 18 minutes le soir
  - A20 : 5 minutes le matin, 3 minutes le soir
- Ces temps varient en fonction de l'heure, et ne dépendent donc pas uniquement des distances
  - Trajet du matin : A16+N8 ou N10+A20 (17 minutes)
  - Trajet du soir : N13 (19 minutes)

Recherche d'un itinéraire optimal via un plus court chemin

- Comment aller le plus rapidement possible d'un point (sommet) A à un point (sommet) B ?
- Algorithme dédié = algorithme de **Dijkstra** :
  - Valide même si critère (à minimiser) = temps de parcours, distance totale, prix des péages, etc.
- Exemples d'applications ? GPS, conception de tables de routage dans les réseaux, ratp.fr, etc.

Problèmes de chemins optimaux

- En fait, plusieurs types de problèmes ont un sens :
  - Plus **court** chemin
  - Plus **long** chemin
    - Problème du plus court chemin avec longueurs ≥ 0 = problème du plus long chemin avec longueurs ≤ 0
- En général, pas d'algorithme efficace connu
  - Quelques cas particuliers pertinents ET « faciles »
    - Plus court chemin avec longueurs ≥ 0 (Dijkstra)
    - Chemin optimal (plus court/long) dans un graphe sans circuit

Ordonnancements optimaux de projets via les graphes (1/2)

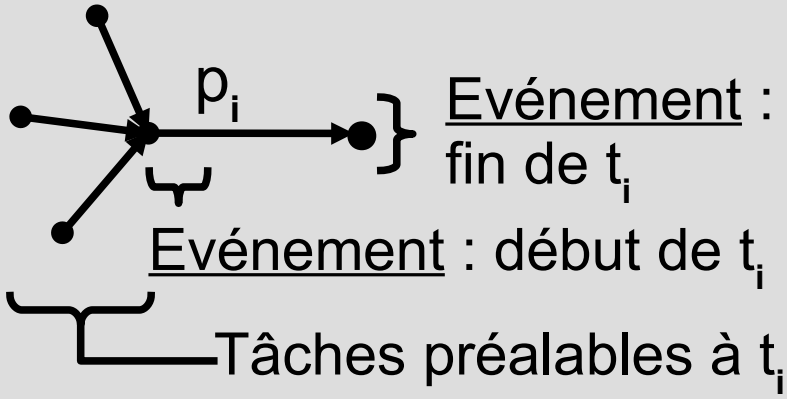
- Soit le problème d'**ordonnancement** suivant :
  - Un projet constitué d'un ensemble de n **tâches**
    - Chaque tâche t_i (ou opération) a une durée d'exécution p_i
  - Pour chaque tâche t_i, on a un ensemble de tâches (qui peut être vide) qui doivent avoir été exécutées avant t_i
    - Contraintes de précédence
      - Tâches préalables, précédence partielle, contraintes de date
  - Planifier les tâches en minimisant la durée du projet ?
    - Contraintes non prises en compte
      - Ressources : supposées suffisantes (ou extensibles)
      - Tâches incompatibles (non exécutables simultanément)

Ordonnancements optimaux de projets via les graphes (2/2)

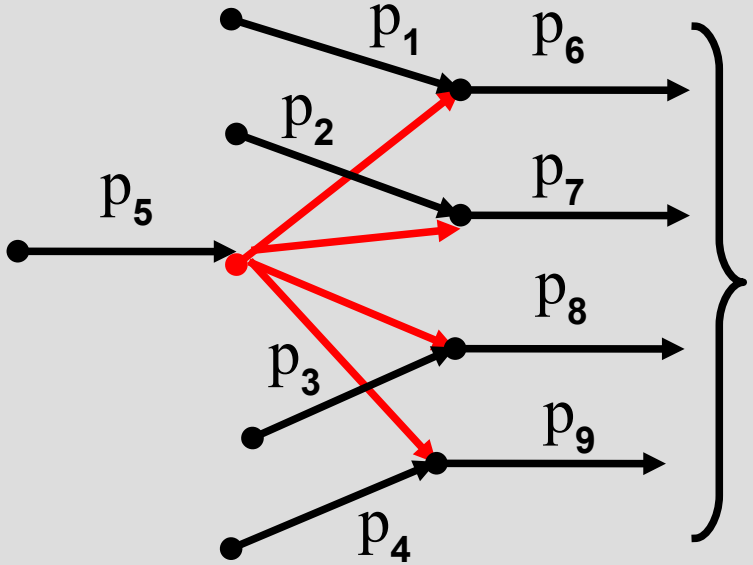
- On peut associer deux modèles à ce problème
  - Modèle (US) PERT (*Project Evaluation and Review Technique*), variante de la CPM (*Critical Path Method*)
  - Modèle MPM (*Méthode des Potentiels Métra*)
    - Méthode française développée par B. Roy, et publiée en 1958 dans sa revue Métra
- Deux modèles, mais des fondements communs
  - Modéliser les tâches et leurs contraintes via un graphe
    - La construction du graphe diffère selon la méthode choisie !
  - Déterminer la durée minimale d'un ordonnancement = calculer **un plus long chemin** dans ce graphe

Description du modèle PERT

- Modèle (ou graphe) dit « événements-tâches »
  - Sommets = « événements »
  - Arcs = tâches



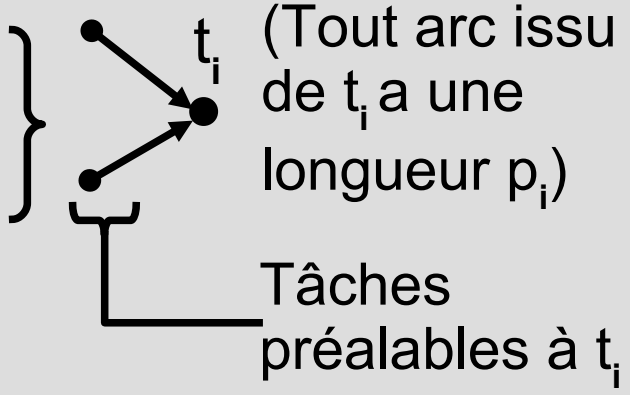
- Inconvénient :
  - Ajout éventuel de nombreuses tâches « fictives » !



Pour tout i<5, la tâche t_{i+5} a comme tâches préalables les tâches t_i et t_5 : pour garantir ces contraintes de précédence, il est donc nécessaire d'ajouter 4 tâches **fictives** de durée d'exécution nulle

Description du modèle MPM

- Modèle (ou graphe) dit « tâches-précédences »
  - Sommets = tâches
  - Arcs = contraintes de précédence



- Avantages :
  - Construction du graphe MPM (à partir des contraintes de précédence) en général simple et évidente
  - Pas de tâches fictives à introduire, sauf 2 (de durée 0)
    - Une tâche « Début », préalable à toute tâche initialement sans tâche préalable (sommet sans prédécesseur)
    - Une tâche « Fin », ayant pour tâche préalable toute tâche qui initialement n'est préalable à aucune autre tâche

Résolution à l'aide de MPM (1/2)

- Comment déterminer une date de début et une date de fin pour chaque tâche, de façon à :
  - respecter les (3 types de) contraintes de précédence,
  - minimiser la durée totale du projet (date de fin) ?
- Possible ssi le MPM obtenu est sans circuit :
  - CN : s'il existe un circuit, on est en présence de contraintes de précédence incompatibles...
  - CS : étant donné un tel graphe, on va relier la durée de l'ordonnancement (**durée du projet**) associé à un **calcul de plus long chemin dans ce graphe**

Résolution à l'aide de MPM (2/2)

- On peut calculer la **date de début au plus tôt** de chaque tâche une fois que celle de chacune de ses tâches préalables est connue (ordre **topologique**)
  - De façon similaire, on peut calculer des **dates de début au plus tard** et des **marges totales**.
- Un **chemin critique** du graphe MPM est un plus long chemin entre « Début » et « Fin » :
  - Les tâches qui le composent sont dites **critiques**,
  - Pour chacune de ces tâches, on a date de début au plus tôt = date de début au plus tard.

Une autre application : flots et trafic dans un réseau ferré

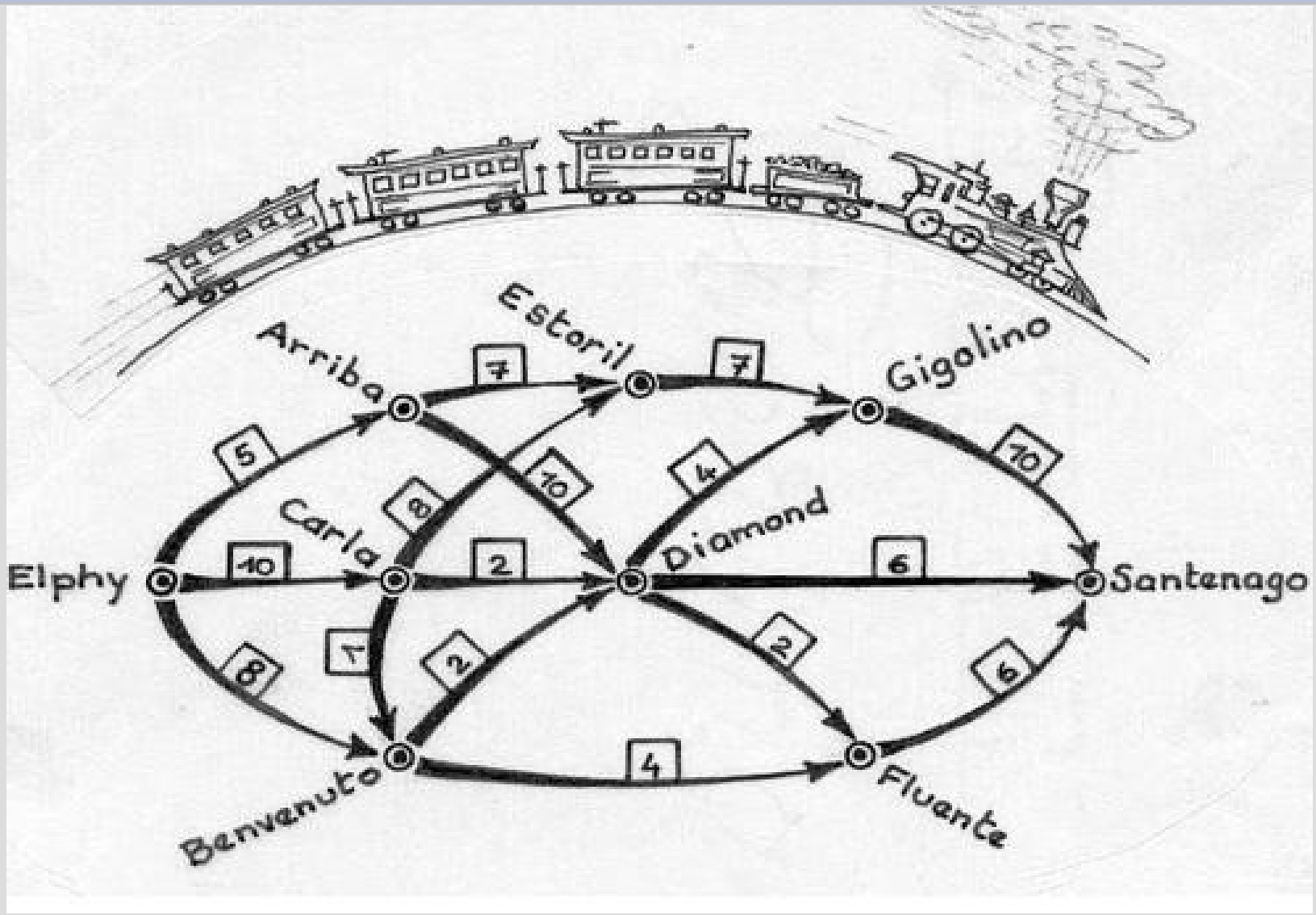
- Idée de base du modèle :
  - Représenter un réseau ferré (ou un réseau de transport) via un graphe (orienté ou non).
  - Chaque nœud/sommet représente une ville/gare.
  - Chaque arc/arête représente un tronçon.
- La quantité de trafic qui circule sur chaque tronçon est représentée par une grandeur appelée **flot**.

Capacité journalière d'un réseau ferroviaire (1/2)

- Sur le réseau ferroviaire, on a indiqué sur chaque tronçon entre 2 villes le nombre maximum de trains qui peuvent passer par jour dans le sens indiqué.
  - Aller de Elphy à Santenago prend moins d'un jour.
  - Chaque jour, il peut partir au plus 23 trains d'Elphy.
- Combien de ces trains, au maximum, peuvent parvenir dans la journée à Santenago ?

(D'après « La vie du rail », 1998.)

Capacité journalière d'un réseau ferroviaire (2/2)



Flot dans un graphe orienté

- Modèle :
  - Un graphe orienté représentant le réseau : un sommet par gare, un arc muni d'une **capacité** par tronçon
  - On part d'une origine (**source**) = Elphy, et on cherche à rallier une destination (**puits**) = Santenago
    - **Réseau de transport** : graphe avec source, puits, capacités
    - Quantité de trains circulant quotidiennement sur un tronçon donné = **flot** sur l'arc associé
    - Toute unité de flot (c'est-à-dire tout train) qui part d'Elphy doit parvenir à Santenago

Formalisation du problème

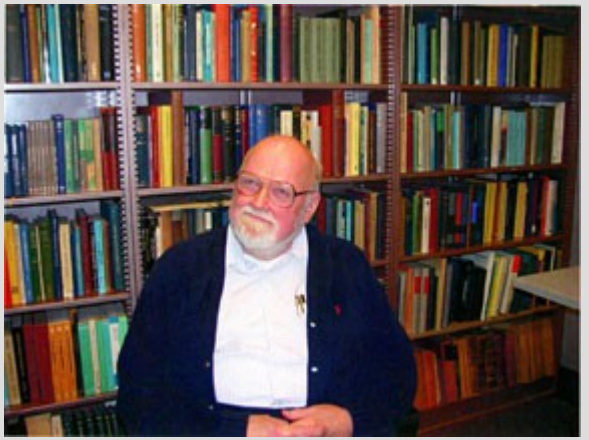
- **Problème du flot maximum** :
  - **Maximiser le flot total** émis par la source (c'est-à-dire le nombre total de trains partant d'Elphy)
  - Tout train parvenant à une gare doit en repartir
    - **Conservation du flot (en chaque sommet)**
  - On doit tenir compte des capacités des tronçons (le flot sur chaque tronçon est borné par sa capacité)
    - **Contraintes de capacité (sur les arcs)**

Est-ce simple de déterminer efficacement un flot maximum ?

- Un arc est **saturé** si son flot = sa capacité
- Notion de flot **complet**
  - Définition : flot dans lequel tout chemin entre la source et le puits contient au moins un arc saturé
    - Comment calculer un flot complet ?
    - Un flot maximum est-il nécessairement complet ?
    - Un flot complet est-il nécessairement maximum ?
  - Calculer un flot complet entre Elphy et Santenago. Quelle est sa valeur ? Peut-on l'améliorer ?

Méthode de Ford et Fulkerson

- Méthode de « marquage » de Ford & Fulkerson pour déterminer un flot maximum (1956)
- Description via la notion de **graphe d'écart** (définie ici à titre informatif)
- Etant donnés un réseau de transport R et un flot f, le graphe d'écart de R associé à f est tel que :
  - Même ensemble de sommets que R
  - Arc (i,j) dans R de flot nul ➔ arc (i,j) de même capacité
  - Arc (i,j) dans R saturé ➔ arc (j,i) de même capacité
  - Sinon, arc (i,j) dans R ➔ 2 arcs : arc (i,j) de capacité $capa_{ij}$-f_{ij}, et arc (j,i) de capacité f_{ij}
- On cherche ensuite un chemin entre la source et le puits dans ce graphe d'écart, et on modifie le flot le long de ce chemin en fonction des capacités



Flot à coût minimum (1/2)

- Variante : un **coût unitaire** de circulation par arc
  - On veut alors acheminer le flot au puits à moindre coût
  - Modèle (graphe) : un réseau (graphe), un coût unitaire et une capacité par arc, une source s, un puits p
- Formalisation : problème du flot à coût minimum
  - Minimiser la somme totale des coûts des arcs
    - Coût d'un arc = quantité de flot sur l'arc fois son coût unitaire
  - Contraintes de capacité
  - Conservation du flot
  - Quantité de flot connue F à acheminer de s à p

Flot à coût minimum (2/2)

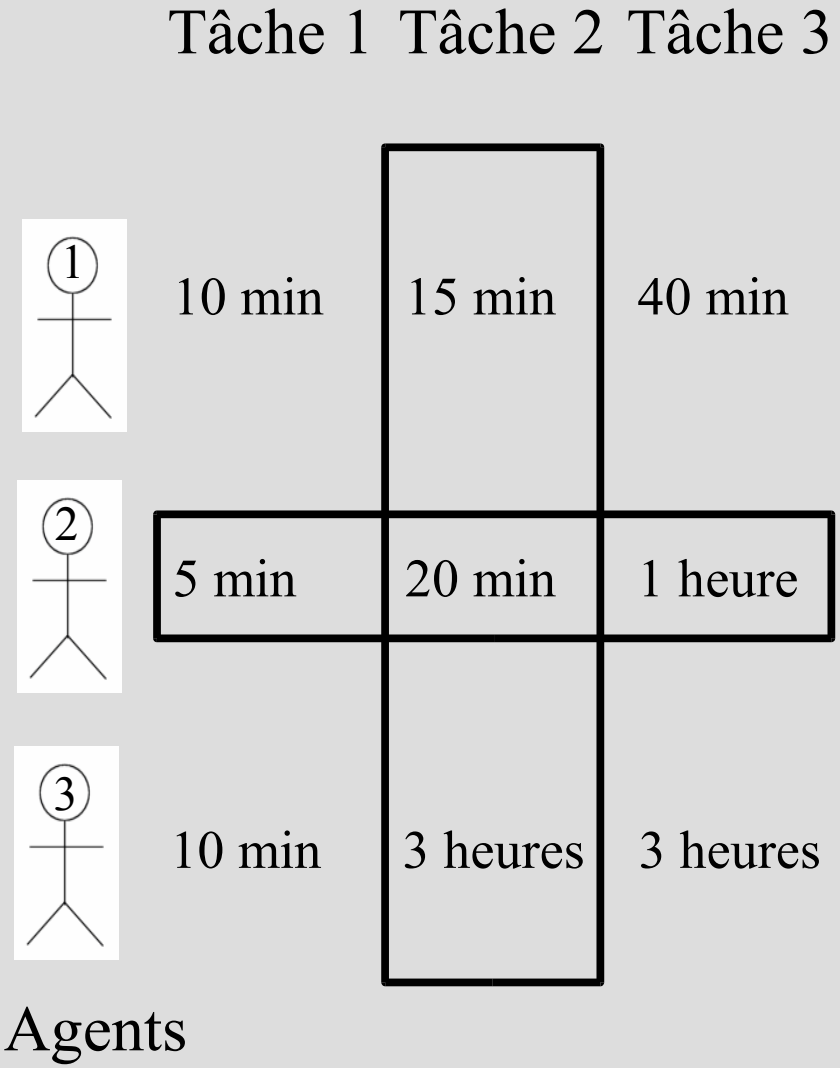
- Beaucoup de similitudes avec d'autres problèmes
  - Si F = valeur d'un flot maximum (cf algorithme de Ford & Fulkerson) : flot maximum à coût minimum
  - Si F = 1 ➔ ?
- Méthode de résolution similaire (graphes d'écart)
  - Algorithme de Busacker & Gowen
  - Chaque arc du graphe d'écart associé à un réseau de transport R et à un flot f a un coût (le coût unitaire de l'arc si celui-ci est dans R, et son opposé sinon)
  - On choisit alors, dans le graphe d'écart, un **plus court chemin** entre la source et le puits

Une 1ère application du flot à coût min. : le frêt (programme de transport)

- Transport de marchandises (entrepôts vers clients)
  - Plusieurs entrepôts, plusieurs clients
  - Chaque entrepôt a une certaine quantité disponible
  - Chaque client a une certaine demande
  - Capacités de transport supposées suffisantes ici
  - Transporter des marchandises d'un entrepôt donné vers un client donné induit un coût unitaire fixe
- Comment satisfaire la demande à moindre coût ?
  - Modèle de flot à coût minimum associé ?

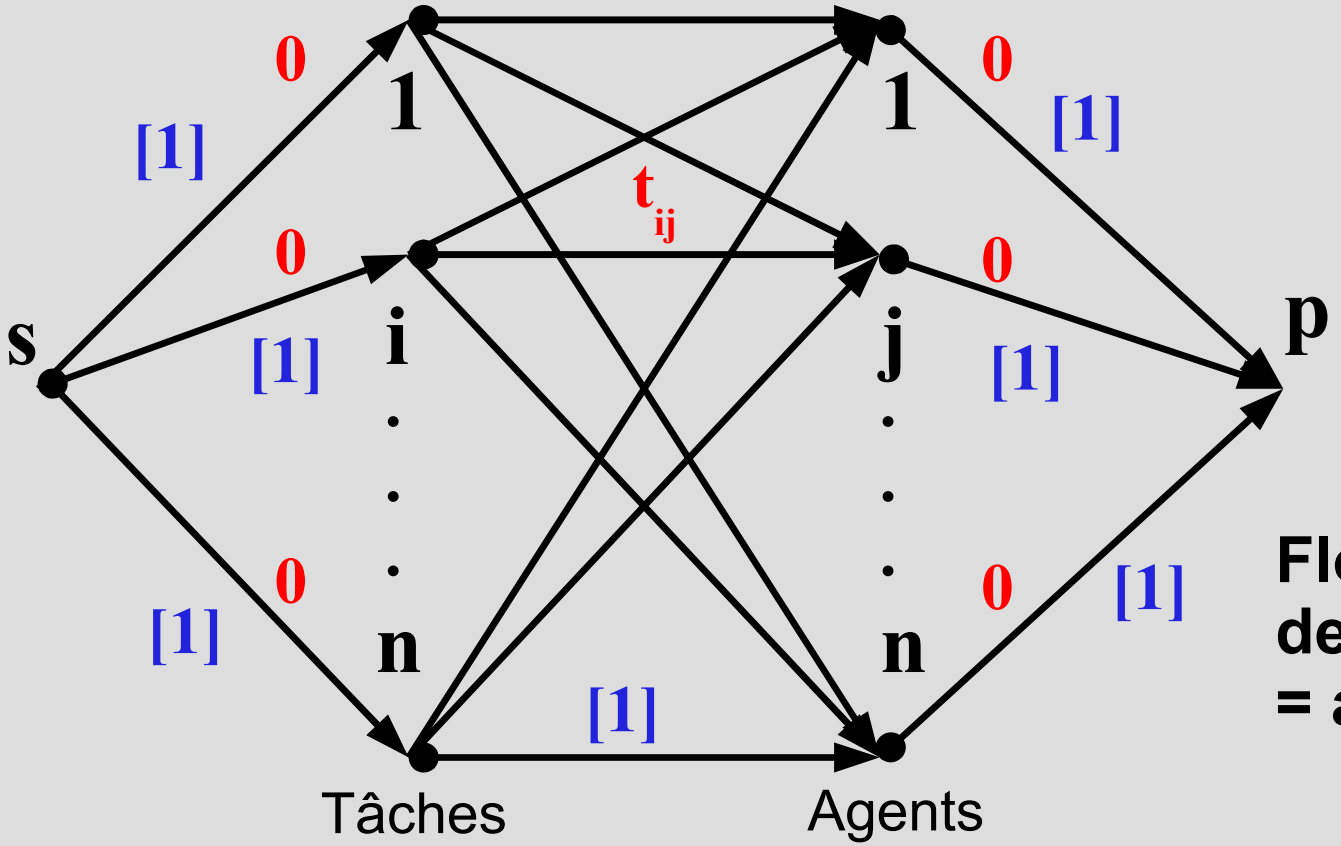
Une 2ème application du flot à coût minimum : l'affectation linéaire

- **Contexte** : dans un entrepôt, n tâches doivent être affectées à n agents de façon à minimiser le temps total (= somme des temps) mis pour effectuer toutes les tâches, sachant que le temps mis par un agent pour effectuer une tâche dépend de la tâche ET de l'agent
- **Formalisation** : chaque paire (tâche, agent) est munie d'une valeur (temps en minutes), et 1 tâche = 1 agent



Calcul d'une affectation linéaire à l'aide d'un flot à coût minimum

- Soit un problème d'affectation linéaire donné par une matrice à n lignes et n colonnes $T=(t_{ij})$:



[capacité]
coût

Flot à coût minimum de valeur n
= affectation optimale !

Bilan

- De nombreux problèmes concrets en lien avec la logistique et le transport peuvent être formalisés via les graphes (cf séance) ou la programmation mathématique (cf prochaine séance) :
  - Résolution numérique (automatisée) via algo. dédiés,
  - Fournit alors des solutions concrètes et exploitables.
- Applications déjà évoquées :
  - Itinéraires optimaux (plus courts chemins),
  - Ordonnancement de tâches et de projets (MPM),
  - Frêt et affectation de ressources (flots).