Corrigé Exercices Semaine 2

Le stade : question 1

- Il s'agit d'un problème classique d'ordonnancement de projet. Ajoutons une tâche fictive de durée nulle qui représente la fin du projet. On considérera que les tâches sont indicées par *i* variant de 1 à *n*, *n* désignant la tâche fictive. On dénote par p_i la durée de la tâche *i*. Pour gérer les précédences entre les tâches, on s'appuie sur un graphe des précédences (ou graphe de projet) *G*=(*X,U*), défini par l'ensemble des tâches *X* et un ensemble d'arcs *U* : un arc (*i*, *j*) signifie que la tâche *i* doit précéder la tâche *j*. Il est facile de construire ce graphe à partir des listes de prédécesseurs du tableau fourni dans l'énoncé. Quant à la tâche fictive de fin du projet, elle suit toutes les tâches sans successeurs.
- Nous avons besoin de variables t_i pour les dates de début des tâches, comptées à partir du temps zéro. Les seules contraintes à respecter sont les contraintes de précédence. Une tâche *j* ne peut démarrer que si toutes les tâches qui doivent la précéder sont terminées, ce qui se traduit par les contraintes (1), appelées *contraintes de potentiels* : s'il existe un arc de *i* à *j*, alors la date de fin de *i* (t_i+p_i) ne doit pas dépasser la date de début de *j*.

$$(1) \quad \forall (i,j) \in U : t_i + p_i \leq t_j$$

- L'objectif à minimiser est la durée du projet, c'est-à-dire la date de début de la tâche fictive. Nous obtenons le modèle mathématique suivant, très compact :

$$(2) \quad \text{Min } t_n$$
$$(1) \quad \forall (i,j) \in U : t_i + p_i \leq t_j$$
$$(3) \quad \forall i = 1 \ldots n : t_i \geq 0$$

Le stade : question 2

- Ce problème est appelé *ordonnancement de projet avec compression des tâches (project crashing)*. Il s'agit de réduire la durée totale de 64 semaines déterminée précédemment. On ajoute des variables s_i pour le nombre de semaines que l'on peut gagner pour chaque tâche *i*. Les contraintes (4) bornent ces variables par les réductions maximales issues du tableau, que nous notons r_i.

$$(4) \quad \forall i \in X : \; s_i \leq r_i$$

- Une variable *A* sert à calculer le nombre de semaines d'avance par rapport au résultat de la question 1. La nouvelle date de fin t_n du projet devra être égale à la précédente échéance *F* diminuée de l'avance, d'où la contrainte (5).

$$(5) \quad t_n = F - A$$

- Par ailleurs, les contraintes (1) devront être modifiées pour tenir compte des variables s_i. La nouvelle date de fin d'une tâche *i* est égale à la date de début de cette tâche, plus sa durée, moins sa réduction, ou encore $t_i + p_i - s_i$. On renomme ces contraintes (1').

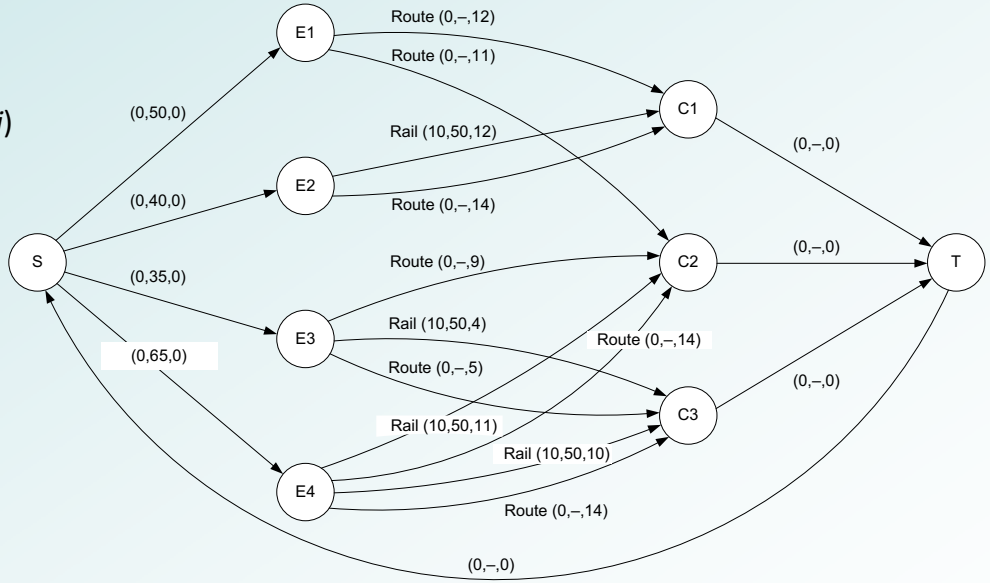
$$(1') \quad \forall (i,j) \in U : \; t_i + p_i - s_i \leq t_j$$

- L'objectif doit aussi être ajusté. Il s'agit maintenant de maximiser le gain que pourrait réaliser le constructeur. Pour chaque semaine d'avance une prime de *P* k€ est versée. En revanche, réduire une tâche *i* d'une semaine coûte c_i k€ (colonne *Coût supplémentaire par semaine* du tableau 6.1). La nouvelle fonction-objectif, notée (2'), est la suivante :

$$(2') \quad \text{Max } P \times A - \sum_{i=1}^{n} c_i \times s_i$$

Rail et Route (1/2)

- Nous allons modéliser ce problème sous forme d'un *problème de flot de coût minimal* avec débit total fixé *F*, défini sur le graphe orienté *G* = (*X*,*U*) de la figure 9.1. L'ensemble *X* des nœuds comprend une couche de nœuds pour les entrepôts et une autre pour les centres. L'ensemble *U* des arcs inclut les liaisons possibles entre entrepôts et centres. Un plan de transport correspond à un flot de *G*, défini par un flux p_{ij} sur chaque arc (*i*,*j*). Un arc (*i*,*j*) est caractérisé par un flux minimal m_{ij} (0, sauf pour les liaisons SNCF), une capacité ou flux maximal K_{ij} (infinie, sauf pour les liaisons SNCF), et un coût de transport par tonne c_{ij}.
- Les deux moyens de transport entre un entrepôt et un centre nécessitent deux arcs parallèles. Un tel graphe, avec au plus *p* arcs dans le même sens entre deux nœuds, est appelé *p-graphe*. Il ne peut pas être codé par une matrice. Par contre, on peut utiliser un codage par liste d'arcs.
- Le graphe ne prend pas en compte les stocks des entrepôts. Pour cela, on crée une *source* (nœud fictif *s*) reliée à chaque nœud entrepôt *i* par un arc (*s*, *i*) de capacité K_{si} égale à la quantité de produit stockée en *i*. Ainsi, le flux quittant l'entrepôt *i* ne pourra excéder cette valeur. Pour faciliter la modélisation, on crée aussi un *puits* (nœud fictif *t*), auquel est relié chaque centre, et un arc de retour (*t*, *s*). Voici le graphe obtenu, avec sur chaque arc (*i* ,*j*) le triplet (m_{ij}, K_{ij}, c_{ij}). Un tiret correspond à une capacité infinie.



Rail ou Route (2/2)

- Le programme linéaire contient les contraintes (2) de conservation du flot ou lois des nœuds ou *lois de Kirchhoff* : la somme des flux arrivant en tout nœud est égale à la somme des flux qui en partent. Cette propriété est vérifiée même pour *s* et *t*, grâce à l'astuce de l'arc de retour. Le flux sur chaque arc vaut au moins m_{ij} (contraintes (3)), sans excéder la capacité maximale K_{ij} (contraintes (4)). La contrainte (5) impose le transport d'une quantité totale F = 180 tonnes, en stipulant un flux égal à F sur l'arc de retour. On pourrait se dispenser de cette contrainte en spécifiant un flux mi
- Il reste à expliquer la fonction-objectif de la ligne (1).
  Comme c_{ij} est un coût par tonne, le coût de passage d'un flux p_{ij} sur un arc (i,j) vaut $c_{ij} \times p_{ij}$. Le coût total de transport, à minimiser, est alors la somme de ces coûts de passage sur l'ensemble des arcs minimal $m_{ts} = F$ dans les données.
- Finalement, grâce à la définition préalable du graphe, on obtient un programme linéaire très compact.
- Notez que les contraintes de positivité des variables de flux sont implicitement assurées par les contraintes (3).

$$(1) \quad \text{Min} \sum_{(i,j)\in U} c_{ij} \times p_{ij}$$

$$(2) \quad \forall i \in X, i \neq s,t : \sum_{(i,j)\in U} p_{ji} = \sum_{(j,i)\in U} p_{ij}$$

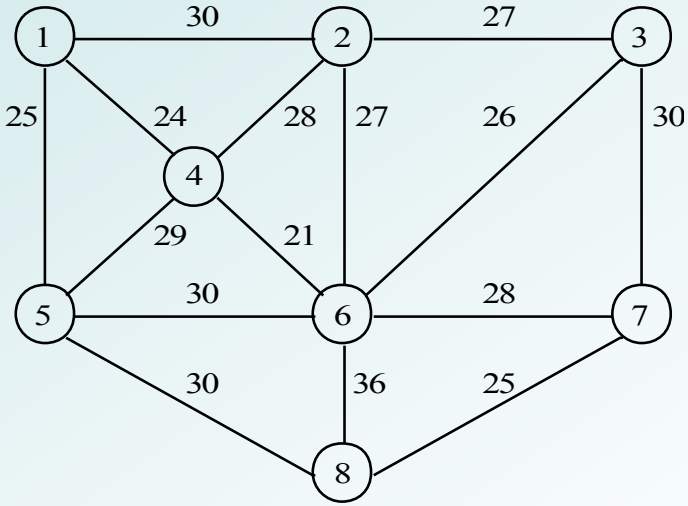
$$(3) \quad \forall (i,j) \in U : p_{ij} \geq m_{ij}$$

$$(4) \quad \forall (i,j) \in U : p_{ij} \leq K_{ij}$$

$$(5) \quad p_{ts} = F$$

Constitution d'équipages (1/2)

- Notons *np* le nombre de pilotes. Ce genre de problème se modélise très bien par un graphe de compatibilité $G = (X,E)$, non orienté. X est un ensemble de *np* nœuds correspondant aux pilotes. Une arête $[i, j]$ existe entre deux nœuds i et j si et seulement si les pilotes i et j sont compatibles, c'est-à-dire s'ils ont une langue et un avion communs pour lesquels ils ont tous deux au moins 10/20. L'arête est valuée par un poids égal au score c_{ij} atteint par l'équipage. La figure suivante donne le graphe résultant, avec les scores pour la question 2. La question travaille sur le même graphe sans les pondération.



Constitution d'équipage (2/2)

- Un ensemble valide d'équipages correspond dans *G* à un ensemble d'arêtes telles que deux quelconques d'entre elles n'ont aucun nœud commun. Un tel ensemble est appelé *couplage* en théorie des graphes. Pour la question 1, on cherche un couplage de *G* de *cardinal maximal*, pour la question 2 un couplage de *poids total maximal*. Le graphe suggère le modèle suivant.
- Pour chaque arête [*i*, *j*] du graphe, une variable binaire indique si cette arête est prise ou pas (3). Les contraintes (2) stipulent que tout nœud *k* a au plus une seule arête incidente. La fonction-objectif (1) cumule les poids des arêtes choisies pour la question 2. Pour le PL de la question 1, on maximise le nombre d'équipages pour voir si tous les pilotes sont pris : il suffit d'enlever c_{ij} de la fonction-objectif. Notez que le couplage de cardinal maximal est un cas particulier de couplage de poids maximal, avec tous les poids égaux à 1.

$$(1) \quad \text{Max} \sum_{[i,j]\in E} c_{ij} x_{ij}$$

$$(2) \quad \forall k \in X : \sum_{[i,j]\in E,\ i=k \text{ ou } j=k} x_{ij} \leq 1$$

$$(3) \quad \forall [i,j] \in E : x_{ij} \in \{0,1\}$$