INFO-F310 - Algorithmique et recherche opérationnelle

Renaud Chicoisne
rchicois@ulb.ac.be

Jérôme De Boeck
jdeboeck@ulb.ac.be

Problème

Le problème de localisation d'installations - ou *Facility Location Problem* (FLP) - est un problème d'optimisation qui consiste à déterminer l'emplacement de centres d'approvisionnement de manière à servir un ensemble de clients de la manière la plus *efficace* possible. La qualité d'un plan de construction est mesurée à partir des coûts de construction des installations ainsi que des coûts de déplacement des clients jusqu'aux centres construits. Dans le cas de la campagne de vaccination en cours, choisir l'emplacement des centres de vaccination pour vacciner le plus rapidement possible une population donnée est un cas particulier de FLP.

Un ensemble de sites J est disponible pour la construction de centres de vaccination. La construction d'un centre au site $j \in J$ a un coût fixe f_j. Chaque famille $i \in I$ doit être affecté à exactement un centre $j \in J$. Un coût de trajet t_{ij} est à considérer dans le cas où la famille i est affectée au centre j. Une famille ne peut se déplacer jusqu'à un centre que si ce dernier est construit. Le coût total d'un choix de sites de vaccination dépend donc des coûts fixes de construction des centres ainsi que des coûts de déplacement des familles.

Chaque famille à une demande d_i de doses de vaccin qui doit être satisfaite par le centre auquel elle est affectée et chaque site de construction disposera d'une capacité maximale de doses c_j (qui doit être considérée comme égale à 0 si aucun centre n'est construit dessus). La demande totale des familles affectées à un centre ne pouvant pas dépasser sa capacité de vaccination, les familles ne peuvent donc pas nécessairement être toutes affectées au centre ouvert qui leur est le plus proche.

Afin d'éviter de trop grandes concentrations de population dans un même secteur, un coût supplémentaire est considéré dans le cas où certains centres proches sont ouverts simultanéments. L'ensemble $\mathcal{S} := \{S_1, ..., S_K\}$ décrit la liste des K sous-ensembles $S_k := \{s_1^k, ..., s_{l(k)}^k\}$ de $l(k)$ sites considérés comme étant proches l'un de l'autre. Pour chaque $S_k \in \mathcal{S}$, si au moins deux sites de S_k sont ouverts, une pénalité p_k est à considérer en plus des coûts de construction et de déplacement. Nous pouvons remarquer que pour chaque sous-ensemble de sites S_k le coût de pénalité est fixe qu'il y ait deux centres proches ouverts ou plus.

1. Instance

Un répertoire contenant des instances `inst-I-J-n.txt` du problème vous est fourni, `I` et `J` représentent les nombres de familles et de sites et `n` l'identificateur de l'instance de cette taille. Chaque fichier est au

format suivant :

$$
\begin{aligned}
&I\ J\ K\\
&f_1 \dots f_J\\
&d_1 \dots d_I\\
&c_1 \dots c_J\\
&t_{11} \dots t_{1J}\\
&t_{21} \dots t_{2J}\\
&\dots\\
&t_{I1} \dots t_{IJ}\\
&s^1_1 \dots s^1_{l(1)}\ p_1\\
&s^2_1 \dots s^2_{l(2)}\ p_2\\
&\dots\\
&s^K_1 \dots s^K_{l(K)}\ p_K,
\end{aligned}
$$

où les K dernières lignes à partir de $s^1_1 \dots s^1_{l(1)}\ p_1$ représentent les ensembles de sites $S_k \in \mathcal{S}$ pour lesquels un coût p_k indiqué en fin de ligne est à considérer si au moins deux sites de cette ligne sont ouverts.

2. Documents à remettre

1. Génération de modèle :
   Un script python3 nommé `generate_model.py` prenant en paramètre en ligne de commande le nom d'une instance dans le même dossier et un paramètre p valant 0 ou 1 qui génère un programme linéaire en nombre entiers de cette instance au format CPLEX LP vu en TP. Ce programme doit être sauvé dans un fichier `model_p_I_J_n.lp` où `p` est le dernier paramètre en ligne de commande et `I`, `J` et `n` dépendent de l'instance. Le script appelé sur l'instance `inst-60-12-0.txt` via la commande

   ```
   python3 generate_model inst-60-12-0.txt 0
   ```

   doit générer un ficher `model_0_60_12_0.lp`. Comme utilisé en TP, le fichier doit pouvoir être résolu et sauver les résultats avec la commande

   ```
   glpsol --lp model_0_60_12_0.lp -o model_0_60_12_0.sol
   ```

   Deux options sont à prévoir en fonction de la valeur du paramètre `p` qui représente le fait d'intégrer ou non une pénalité. Si `p` = 0, aucune pénalité n'est à considérer si plusieurs centres proches sont ouverts simultanément et les dernières lignes des fichiers d'instance après les coûts de déplacement peuvent être ignorés. Si `p` =1, le modèle généré devra considérer des pénalités si plusieurs centres proches sont ouverts tel que décrit dans le problème.
2. Programme linéaire :
   Deux fichiers `model_0_60_12_0.lp` et `model_1_60_12_0.lp` retourné par la fonction du point précédent sur l'instance `inst-60-12-0.txt` via les commandes

   ```
   python3 generate_model inst-60-12-0.txt 0
   python3 generate_model inst-60-12-0.txt 1
   ```

3. Solution du programme linéaire :
   Deux fichiers `model_0_60_12_0.sol` et `model_1_60_12_0.sol` qui contiennent les solutions obtenues en considérant ou non la pénalité, de l'instance `inst-60-12-0.txt`.
4. Rapport
   Un rapport LaTeXcompilé au format `pdf` qui détaille votre formulation et ses résultats. Le rapport doit contenir au minimum :
   — vos noms, prénoms et matricules,
   — le système d'exploitation de la machine sur laquelle vous avez fait vos tests,
   — le programme linéaire utilisé en décrivant les différentes notations comme en TP,

— expliquer clairement pourquoi les variables, contraintes et fonction objectif du programme modélisent entièrement le problème
— une analyse des résultats de la résolution des différentes instances. Cette analyse peut comparer les objectifs et temps de résolution (attention à bien noter ceux-ci à la fin d'une résolution) en fonction des différentes instances, de leur taille et de la considération ou non de la pénalité en cas d'ouverture de centres proches. Tâchez d'interpréter le pourquoi des résultats.

3. Consignes de remise

Ce travail est à faire par groupe de 2 personnes. Vous êtes invité à me communiquer vos groupes pour le 19 avril (un email par groupe reprenant les noms et matricules suffit). Si vous ne trouvez personne, vous pouvez m'envoyer un email avant cette date. Si vous ne vous manifestez pas avant le 19 avril, je ne pourrai vous garantir de trouver un partenaire pour le projet.

Veillez à scrupuleusement respecter le nom des fichiers remis et la syntaxe des appels en ligne de commande. Le rapport doit être remis au format `pdf`.

Pour remettre votre projet sur l'UV, nous vous demandons de
— créer localement sur votre machine un répertoire de la forme `NOM1_NOM2` (exemple :`CHICOISNE_DEBOECK`, sans espace) dans lequel vous mettez les fichiers demandés (sans y inclure de fichier de données)
— compresser ce répertoire via un utilitaire d'archivage produisant un `.zip` (aucun autre format de compression n'est accepté)
— de soumettre le fichier archive `.zip`, et uniquement ce fichier, sur l'UV

Le projet est à remettre pour le 19 avril 2021 à 14h sur l'UV. Tout manquement aux consignes ou retard sera sanctionné directement d'un **0/10**.