INFO-F310 - Algorithmique et recherche opérationnelle

Jérôme De Boeck
Jerome.De.Boeck@ulb.be

1. Problème de flot maximum

Le problème de flot maximum consiste à envoyer un maximum de ressources d'un noeud s à un noeud t dans un graphe $G = (V, A)$ où chaque arc $ij \in A$ a une capacité maximum c_{ij}. Ce problème peut être modélisé et résolu à l'aide d'un programme linéaire mais également par la méthode des *chemins augmentants* vue en cours.

Nous vous demandons de résoudre le problème de flot maximum avec ces deux méthodes :

— Formulez le problème de flot maximum à l'aide d'un programme linéaire et résolvez-le avec le solver `glpk`.

— Implémentez la méthode des chemins augmentant en `python3`.

Les formulations et méthodes utilisées devront être décrites dans un rapport qui présentera également de l'efficacité des deux algorithmes. Pour les deux approches, vous pouvez limiter les temps de calcul à 5 minutes. Votre rapport doit contenir les résultats des deux méthodes et analyser l'impact de la taille des instances dans les temps de résolution. Pour les instances résolues avec les deux méthodes en moins de 5 minutes, vérifiez que les solutions trouvés ont la même valeur et comparez les temps de résolutions. Pour les autres instances, indiquez les instances les plus grands résolues avec au moins une méthode.

2. Instances

Les instances sont disponibles sur l'UV. Chaque fichier est nommé `inst-n-p.txt` où n est le nombre de noeuds de l'instance et p la densité du graphe ($\frac{|A|}{|V|^2}$). Le format d'un fichier est le suivant :

`nodes` n `(nombre de noeuds)`
`source` u `(numéro de la source)`
`sink` v `(numéro du puit)`
`arcs` a `(nombre d'arcs)`
i j c `(présence d'un arc` ij `et capacité` c_{ij} `associée)`
...

Un nombre exhaustif d'instances est donné. Toutes ne doivent pas être résolues et rapportées dans le rapport (certaines seront beaucoup trop grands pour glpk en 5 minutes).

3. 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 `inst-n-p.txt` dans le même dossier et 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-n-p.lp`. Le script appelé sur l'instance `inst-300-0.3.txt` via la commande

   ```
   python3 generate_model.py inst-300-0.3.txt
   ```

   doit générer un ficher `model-300-0.3.lp`. Comme utilisé en TP, le fichier doit pouvoir être résolu et sauver les résultats avec la commande

   ```
   glpsol --lp model-300-0.3.lp -o model-300-0.3.sol
   ```

2. Programme linéaire :
   Un fichier `model-300-0.3.sol` retourné par la fonction du point précédent sur l'instance `inst-300-0.3.txt` via la commande

   `python3 generate_model.py inst-n-p.txt`

3. Solution du programme linéaire :
   Un fichier `model-300-0.3.sol` qui contient la solution optimale obtenue en considérant l'instance `inst-300-0.3.txt`.

4. Méthode des chemin augmentant :
   Un script python3 nommé `chemin_augmentant.py` prenant en paramètre en ligne de commande le nom d'une instance `inst-n-p.txt` dans le même dossier qui trouve la solution optimale et qui la stock dans un fichier `model-n-p.path`.

5. 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,
   — la formulation utilisée en décrivant les différentes notations comme en TP. Expliquez en détail pourquoi les variables, contraintes et fonction objectif du programme modélisent entièrement le problème, tâchez d'avoir une formulation de qualité.
   — une description de la méthode des chemins augmentant implémentée,
   — une analyse des résultats de la résolution des différentes instances. Cette analyse peut comparer les temps de résolution pour des instances de taille croissante pour les deux méthodes de résolution. Vous pouvez analyser les temps de résolution moyens ainsi que l'écart-type pour les différentes tailles d'instances. Tâchez d'interpréter le pourquoi des résultats et identifiez si une méthode semble plus appropriée.
   Un nombre exhaustif d'instances est fournies. Le rapport ne doit pas contenir les résultats de toutes les instances. Sélectionnez les instances afin d'être le plus complet dans vos résultats.

4. Consignes de remise

Ce travail est à faire par groupe de 2 personnes. Vous êtes invités à communiquer vos groupes pour le 17 avril à l'adresse `Jerome.De.Boeck@ulb.be` (un email par groupe reprenant **les noms et matricules** suffit). Si vous ne trouvez pas de partenaire, envoyez également un email avant cette date. Toute personne ne s'étant pas manifestée pour le 17 avril ne sera pas considérée dans 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 :`FORTZ_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 (une seule remise par binôme)

Le projet est à remettre pour le 22 mai à 14h sur l'UV. Tout manquement aux consignes ou retard sera sanctionné directement d'un **0/10**.