Master 1 - Spécialité Androide Année 2018/2019

Module RP
Résolution de Problèmes

Projet

Autour du Google Hash Code 2019 "Photo slideshow"

Le "Google Hash Code" est une compétition de programmation pour des équipes d'étudiants ou de non-étudiants. La compétition est basée sur un problème, le plus souvent en optimisation combinatoire, et sur des instances de taille importante à résoudre. Le gagnant de cette compétition est celui qui a le meilleur "score", c'est-à-dire une solution optimisant la fonction objectif sur ces instances.

Pour ce projet du module "Résolution de Problèmes", nous vous proposons un problème directement issu du "Qualification Round" du Google Hash Code 2019.

On considère dans ce projet la résolution d'un problème de choix d'ordre de passage de photos dans une présentation vidéo "a photo slideshow" en anglais. Pour cela, nous allons utiliser des techniques de résolution heuristique et exacte. Vous pouvez trouver la version du projet à la page

`https://codingcompetitions.withgoogle.com/hashcode/archive`

sous la forme d'un sujet `hashcode2019_qualification_task.pdf` et d'une archive de plusieurs instances appelée `qualification_round_2019.in.zip`.

La page web de ce projet est

`http://www.lip6.fr/Pierre.Fouilhoux/documentens.php`

où seront mis en ligne toutes les informations et indications utiles, ainsi que l'affichage des meilleurs scores obtenus lors du projet !

1 Définition du problème

L'énoncé suivant reprend celui du sujet initial du Google HashCode en introduisant certaines notations utiles nouvelles.

1.1 Enoncé du problème

Données :
On considère un ensemble de N photos. Chaque photo $i \in \{1, \ldots, N\}$ est caractérisée par :
- une orientation : horizontale (H) ou verticale (V)
- un nombre M_i de mots-clé (tags en anglais) : $1 \leq M_i \leq 100$
- une liste de M_i mots-clé.

Un mot-clé est une chaîne de 1 à 10 caractères minucsules ASCII, sans espace.

Sortie :
On veut construire une présentation des photos en vignettes (photo slideshow, en anglais). Une présentation est définie comme une liste ordonnée d'au moins une vignette.
Une vignette (slide, en anglais) contient :
- soit une unique photo horizontale,
- soit exactement deux photos verticales (mises côte à côte).

1.2 Score d'une solution

Toute la difficulté de ce problème repose sur la façon dont est calculé le "score" associé à une présentation. Le but de ce score est d'inciter à ce que, à la fois, l'enchaînement de deux vignettes possède une certaine logique mais aussi à ce que deux vignettes successives ne soient pas trop similaires.
On peut imaginer qu'une telle présentation soit diffusée pour animer par exemple une salle d'attente (aéroport, hôpital,...) ou pour présenter les plus belles photos d'un artiste sur un site internet.

La liste des mots-clé d'une vignette est
- soit la liste des mots-clé de son unique photo horizontale
- soit l'union (sans doublon) des deux listes de ses deux photos verticales.

Dans une présentation, $S = (S_1, \ldots, S_k)$, on appelle transition le passage d'une vignette S_i à la vignette suivante S_{i+1}.
Le score d'une transition entre deux vignettes successives S_i et S_{i+1} est le minimum entre :
- le nombre de mots-clé communs entre S_i etS_{i+1}
- le nombre de mots-clé apparaissant dans S_i et non dans S_{i+1}
- le nombre de mots-clé apparaissant dans S_{i+1} et non dans S_i.

Le score d'une présentation de k vignettes est la somme des $k-1$ transitions entre ses vignettes. (Une présentation d'une unique vignette a donc un score nul).

1.3 Format des fichiers

Le fichier archive contient plusieurs instances du concours Google Hash Code. Une instance de description `a_example.txt`, une instance test `c_memorable_moments.txt`, et 3 instances de 80000 à 90000 photos.
Voici l'instance `a_example.txt` qui permet de décrire le format des instances :

```
4
H 3 cat beach sun
V 2 selfie smile
V 2 garden selfie
H 2 garden cat
```

On peut lire que cette instance contient 4 photos. La première est une photo horizontale avec une liste de 3 mots-clé qui sont l'ensemble $\{cat, beach, sun\}$. Notez que pour la suite, cette première photo sera dite d'indice 0. La deuxième est une photo verticale qui sera d'indice 1 etc.

En retour, vous devrez produire une présentation dans un fichier txt d'extension .sol, qui doit respecter le format très simple. Ce format est indiqué ici par l'exemple d'une présentation qui est valide pour l'instance `a_example.txt` précédente.

```
3
0
3
2 1
```

Ce format correspond à une présentation de 3 vignettes. La première vignettre contient une photo H qui est celle de numéro 0 dans le fichier `a_example.txt`. La deuxième vignette contient une photo H de numéro 3. La troisième vignette contient deux photos V de numéro 2 et 1 (rappelez-vous que l'ordre des deux photos V dans une vignette n'a pas d'importance).

2 Entrées-Sorties

Exercice 1 **: Entrées-Sorties**

Q 1.1. Récupérer le code `C++` fourni pour le projet. Vous pouvez le compiler sous linux en lançant la commande `make` dans le répertoire. Il vous permettra d'obtenir le score pour vos fichiers solutions.

Vous pouvez choisir librement votre langage pour le projet. Faites attention toutefois à choisir un langage adapté.

Q 1.2. Créer des fonctions lisant p% d'une instance de n photos : c'est-à-dire lisant exactement $\lfloor \frac{p \times n}{100} \rfloor$ des lignes du fichier (où $\lfloor a \rfloor$ est la partir entière inférieure de a).

Q 1.3. Créer une structure de données simple (tableau) permettant de stocker l'instance. Il peut être utile de stocker à part les photos H et les photos V mais sans perdre leur numéro d'ordre issu du fichier instance.

Q 1.4. Faire une fonction permettant d'obtenir la présentation simple suivante : toutes les photos H en premier dans l'ordre de l'instance, puis toutes les photos V par lot de 2 dans l'ordre de l'instance. (Notez que s'il y a un nombre impair de photos V, l'une d'entre elles ne peut être mise dans la présentation).

Q 1.5. Faire des fonctions permettant d'écrire sur disque une présentation au format désiré.

Q 1.6. Faire une fonction d'évaluation permettant d'avoir le score d'une présentation.

Q 1.7. Vérifiez que votre ficher solution a bien le même score selon votre code et selon le code fourni `Check_RP` pour le projet.

3 Complexité du problème

Exercice 2 **: Complexité**

Prouvez que ce problème est NP-difficile même quand les instances ne possèdent que des photos horizontales.

4 Techniques de résolutions

Le projet consiste à étudier différentes techniques de résolutions pour le problème. Vous pouvez utiliser le langage de programmation et un solveur PLNE de votre choix [1].
Vous comparerez ces méthodes dans une partie expérimentale, vous pouvez lire rapidement le document suivant qui résume les approches en résolution exacte ou à garantie expérimentale [2].
Tout au long de ce projet, vous allez devoir faire des compromis entre tailles d'instances résolues et temps de calcul. En effet, les instances fournies par Google Hash Code sont très grandes. A chaque méthode utilisée, évaluez votre méthode sur des **instances de tailles croissantes** en lisant un pourcentage donné de l'instance.

Exercice 3 : Méthodes gloutonnes

Q 3.1. Implémenter la méthode gloutonne suivante :
- Commencer par une vignette contenant une photo H ou un couple de photos V choisies au hasard
- Ajouter itérativement après la vignette S_i une vignette S_{i+1} constituée d'une photo H ou V (non sélectionnée jusqu'ici) maximisant la transition. S'il s'agit d'une photo V, compléter ensuite la vignette en choisissant une photo V maximisant la transition.

Noter que cette méthode est trop lente pour résoudre des instances de grande taille. Proposer une méthode gloutonne plus rapide (mais peut-être moins performante).

Exercice 4 : Méta-heuristiques

Dans cette partie, on désire mettre en oeuvre des méthodes méta-heuristiques pour le problème.

Q 4.1. Implémenter la méthode de descente stochastique à partir de votre solution gloutonne à partir des voisinages suivants :

a - Permuter deux vignettes

b - Permuter une des deux photos V entre deux vignettes

A chaque itération de la descente, vous tirerez au sort quel voisinage appliquer ;

Q 4.2. Proposer des améliorations de la méthode de descente proposée ci-dessous : ce peut-être par exemple d'autres voisinages, d'autres probabilités, un cadre différent de méta-heuristiques : recuit simulé, méthode tabou ou algorithme génétique,...

1. Dans le concours Google Hash Code, seuls les solveurs publiques (libres ou open-source) peuvent être utilisés
2. Module MAOA `http://www.lip6.fr/Pierre.Fouilhoux/MAOA` Rappel de cours "Résolution exacte ou à garantie expérimentale'

Exercice 5 : Formulations PLNE

Plusieurs formulations en programmes linéaires en nombres entiers (PLNE) peuvent être obtenues pour ce problème. Nous allons nous concentrer sur des formulations compactes, c'est-à-dire possédant un nombre polynomial de variables et d'inégalités.

Pour simplifier les deux sections suivantes, nous allons uniquement considérer l'instance `b_lovely_landscape.txt` qui ne possède que des photos horizontales.

Q 5.1. En vous inspirant de l'annexe qui propose des formulations pour le problème du voyageur de commerce, proposez une formulation PLNE pour le problème dans le cas où toutes les photos sont horizontales.

Q 5.2. En utilisant un solveur de votre choix (gurobi, cplex, xpress, glpk), utiliser cette formulation pour résoudre les instances du problème.

Exercice 6 : Heuristiques d'arrondi

Q 6.1. Implémenter une méthode d'arrondi pour le problème à partir des relaxations linéaires de la formulation entière de l'exercice précédent :

- Trier les variables y_{ij} par ordre décroissant de valeur
- Choisir la transition y_{ij} selon cet ordre si elle ne conduit pas à former un cycle.

Q 6.2. Proposer une autre méthode d'arrondi. Cela peut-être une amélioration de la méthode précédente ou un couplage entre différentes méthodes déjà rencontrées.

Exercice 7 : Analyse expérimentale

Comparer les différentes méthodes des exercices précédents (capacités, rapidité). Déduisez quelles méthodes sont les plus adaptées pour résoudre les instances pour des valeurs de p croissantes.

Exercice 8 : A vous de jouer

Plusieurs pistes bonus sont à votre disposition.

Q 8.1. Pour les instances difficiles à résoudre par les méthodes précédentes, proposer VOTRE méthode... La meilleure méthode aura des points bonus !

Q 8.2. Pour les instances possédant des photos V, proposez une formulation PLNE efficace.

5 Documents à fournir

Pour le rendu final, vous rédigerez un rapport très synthétique contenant la description de votre travail, c'est-à-dire :
- une description synthétique de vos méthodes

- une descrition rapide de votre code et de la façon de l'utiliser
- vos analyses pratiques et théoriques
- et vos meilleurs résultats !
- joignez également votre code (commenté bien sûr)

Envoyer un mail à Evripidis.Bampis@lip6.fr et à Pierre.Fouilhoux@lip6.fr avant la séance prévue pour le rendu.
Votre mail doit comporter :
- un objet commençant par [RP]
- contenir 2 fichiers dont le nom commencent par nom1_nom2 en mettant les noms de votre binome par ordre alphabétique
- un des fichiers est l'archive tar.gz de votre code
- un des fichiers est votre rapport en pdf.

Dates du projet

Sujet distribué par mail la semaine du **25 mars 2019**.
Une mini-soutenance aura lieu la semaine du **6 mai 2019**.

Annexe : Formulations compactes pour le problème du voyageur de commerce asymétriques

On considère le problème du voyageur de commerce asymétrique.
Données : n villes avec c_{ij} coût de transport de i à j (asymétrique).
Objectif : Déterminer un circuit passant par toutes les villes de plus petit coût.

Noter qu'un problème symétrique peut toujours être vu comme asymétrique.

On se propose d'utiliser les variables suivantes :
$x_{ij} = 1$ si l'arc (i, j) est dans le circuit et 0 sinon.
On appelle *sous-tour* le fait qu'au lieu d'avoir un unique tour passant par tous les sommets, la solution décrite par les variables x est composée de plusieurs tours.
Dans la formulation PLNE "cadre", la contrainte (1) a pour but d'interdire toute solution possédant des sous-tours.

$$\begin{aligned}
\text{Min } & \sum_{i,j} c_{ij} x_{ij} & \\
& \sum_{j \in V} x_{ij} = 1 & \forall i \in V, \\
& \sum_{i \in V} x_{ij} = 1 & \forall j \in V, \\
& \text{<Elimination des sous-tours>} & \qquad (1) \\
& x_{ij} \in I\!N & \forall i \in V, j \in V \setminus \{i\}).
\end{aligned}$$

On peut ajouter également les inégalités $x_{ij} + x_{ji} \leq 1$.

Cette formulation PLNE "cadre" est alors valide car s'il n'y a qu'un seul tour et comme chaque sommet n'a qu'un unique arc sortant, la solution est bien un tour passant par tous les sommets.
Cette annexe propose trois façons d'implémenter cette contrainte de sous-tours.

• *Elimination des sous-tours" par la formulation MTZ*

Formulation de Miller-Tucker-Zemlin (MTZ).
Ajout des variables réelles u_i, $i = 1, ..., n$, associées aux villes
Ajout des contraintes :

$$\begin{aligned}
u_1 &= 1, & \\
2 \leq u_i &\leq n & \forall i \in V \setminus \{1\}, \\
u_i - u_j + 1 &\leq n(1 - x_{ij}) & \forall i \in V \setminus \{1\}, j \in V \setminus \{1, i\}.
\end{aligned}$$

On appelle ces dernière inégalités *inégalités MTZ.*
Cette formulation possède seulement n variables (continues) de plus. En revanche, c'est

une très très mauvaise formulation car elle possède une mauvaise valeur de relaxation linéaire.

• *Elimination des sous-tours" par les flots*

Ajout des variables de flots réelles z_{ij}, pour tout $i \in V$, j$\in V \setminus \{1, i\}$.
Ajout des contraintes :

$$\begin{aligned}
\sum_{j \in V \setminus \{1\}} z_{1j} &= |V| - 1, \\
\sum_{j \in V \setminus \{1,i\}} z_{ij} + 1 &= \sum_{j \in V \setminus \{i\}} z_{ji} && \forall i \in V \setminus \{1\}, \\
z_{ij} + z_{ji} &\leq (|V| - 1)(x_{ij} + x_{ji}) && \forall i \in V, j \in V \setminus \{1, i\} \\
z_{ij} &\geq 0 && \forall i \in V, j \in V \setminus \{1, i\}.
\end{aligned}$$

On peut remarquer que les variables z portent un flot de valeur $|V| - 1$ sortant du sommet 1, mais qu'il n'y a pas d'inégalité de flot entrant au sommet 1.
Cette formulation possède n^2 variables (continues) de plus. Sa valeur de relaxation linéaire est bien meilleure que celle utilisant MTZ.

• *"Elimination des sous-tours" par des flots désagrégés*

Ajout de $|V|$ ensemble de variables de flots réelles z^k_{ij}, pour tout $i \in V$, $j \in V \setminus \{1, i\}$ et pour tout $k \in V \setminus \{1\}$.
Ajout des contraintes :

$$\begin{aligned}
\sum_{j \in V \setminus \{1\}} z^k_{1j} &= 1 && \forall k \in V \setminus \{1\} \\
\sum_{j \in V \setminus \{1,i\}} z^k_{ij} &= \sum_{j \in V \setminus \{i\}} z^k_{ji} && \forall k \in V \setminus \{1\}, \forall i \in V \setminus \{1, k\}, \\
\sum_{j \in V \setminus \{1,k\}} z^k_{kj} + 1 &= \sum_{j \in V \setminus \{k\}} z^k_{jk} && \forall k \in V \setminus \{1\}, \\
z^k_{ij} + z^k_{ji} &\leq x_{ij} + x_{ji} && \forall i \in V, j \in V \setminus \{i, 1\}, \forall k \in V \setminus \{1\} \\
z^k_{ij} &\geq 0 && \forall i \in V, j \in V \setminus \{1, i\}, \forall k \in V \setminus \{1\}.
\end{aligned}$$

On peut remarquer que chaque ensemble de variables z^k porte un flot de valeur 1 sortant du sommet 1, mais qu'il n'y pas d'inégalité de flot entrant au sommet 1.
Cette formulation a une très bonne valeur de relaxation. Toutefois elle possède n^3 variables (continues) supplémentaires ainsi qu'un nombre d'inégalité en n^2...

Il existe d'autres formulations non compactes pour le problème de voyageurs de commerce qui sont plus complexes à mettre en œuvre mais qui sont davantage efficaces.