MTH6412A : Projet voyageur de commerce (phase 5)

Lundi 23 novembre 2020

[Dominique Orban]

Pondération: 10%, travail en équipes de deux

Remise: le 7 décembre 2020

Objectif : application

La dernière partie du projet consiste à mettre votre méthode de résolution du
TSP en oeuvre afin de reconstruire des images déchiquetées.
Téléchargez le fichier `shredder.zip` disponible sur le site Moodle du cours.
Celui-ci contient plusieurs répertoires :

- `bin` contient quelques outils Julia ;
- `images/original` contient plusieurs images originales au format PNG ;
- `images/shuffled` contient les mêmes images, passées à la déchiqueteuse.

Les images "déchiquetées" ont été créés en découpant chaque image en un certain nombre de bandes verticales et en réordonnant ces bandes dans un ordre aléatoire. Les bandes sont de largeur constante.

Le but de cette phase est de reconstruire les images le mieux possible.
Cela peut se faire à l'aide du TSP en imaginant que chaque bande verticale représente un noeud du graphe complet, et le poids de chaque arête est une mesure de dissimilarité entre deux bandes verticales. L'idée consiste à supposer que dans l'image originale, deux bandes adjacentes sont très *semblables*, i.e., très peu *dissemblables*.
On cherche ainsi un chemin simple de longueur maximale à travers les noeuds du graphe, qui est aussi de poids minimal.
Comme nous l'avons vu en classe, ce problème est équivalent au TSP.

Pour formuler le problème comme un TSP, un noeud fictif (numéro zéro) a été ajouté ainsi qu'une arête de poids nul reliant ce noeud fictif à chaque autre noeud du graphe. En cherchant une tournée

minimale sur ce graphe modifié, puis en retirant le noeud zéro, nous obtenons le chemin cherché.

Modules Julia

Vous aurez besoin d'installer les modules Julia suivants :

- `FileIO`
- `Images`
- `ImageView`
- `ImageMagick`

Méthode

Pour chaque image déchiquetée, un fichier `tsp` a déjà été créé pour vous dans le répertoire `tsp/instances`.
Ces fichiers `tsp` doivent être lus avec votre version mise à jour de `read_stsp.jl` (qui lit les poids des arêtes).
Le champ `data` des noeuds sera simplement l'indice du noeud.

De façon alternative, vous pouvez lire vous-même une image déchiquetée et obtenir la dissimilarité entre chaque paire de colonnes à l'aide des commandes ::

```
picture = load(input_name)
nb_row, nb_col = size(picture)
w = zeros(nb_col, nb_col)
for j1 = 1 : nb_col
  for j2 = j1 + 1 : nb_col
    w[j1, j2] = compareColumn(picture[:, j1], picture[:, j2])
  end
end
```

Une fois une tournée identifiée, construisez une liste des noeuds le long de cette tournée *sans retirer le noeud zéro*.
À l'aide de la fonction `write_tour()`, créez un fichier `.tour` au format TSPLib qui décrit votre tournée.
Des exemples de fichiers `.tour` sont disponibles dans le répertoire `tsp/tours` ; ceux-ci ont été identifiés par une méthode de résolution du TSP, mais ne donnent pas nécessairement une solution optimale.

Étant donné un fichier `.tour`, on obtient l'image reconstruite correspondante à l'aide de la fonction `reconstruct_picture()`.
Cette fonction suppose que la tournée donnée par le fichier `.tour` débute au noeud 0.

À vous de créer vos fichiers `.tour` à l'aide de votre implémentation de RSL et HK et de générer les images reconstruites correspondantes.
Jouez sur tous les paramètres que vous désirez pour essayer d'obtenir la meilleure reconstruction possible.

Pour chaque image reconstruite, donnez la longueur de la meilleure tournée trouvée, l'image originale ainsi que l'image reconstruite côte-à-côte.

Références

Le code utilisé dans cette partie du projet provient de https://github.com/robinhouston/image-unshredding.
Je vous propose de parcourir ce site web pour avoir un peu plus d'information.
Les fichiers situés sous `tsp/tours` ont été générés par la librairie LKH, une implémentation de l'heuristique de Lin et Kernighan décrite à l'adresse http://webhotel4.ruc.dk/~keld/research/LKH.

Directives

- Écrire du code **lisible**, aéré, documenté et commenté. On pourra se reporter aux lignes directrices pour la rédaction de code Julia : https://docs.julialang.org/en/v1/manual/style-guide
- Vos méthodes doivent être documentées suivant le schéma donné dans la documentation officielle de Julia : https://docs.julialang.org/en/v1/manual/documentation
- Déposer votre rapport sous forme de carnet Pluton au format **PDF** sur Moodle et consigner la version julia **jl** sur la branche *phaseX* de votre fork.
  Des cellules Markdown doivent guider le lecteur à travers votre rapport.