

TP Optimisation – session 4 – Algorithmes d’approximation

1^{er} janvier 2025

Objectif d’apprentissage : cette séance vise à concevoir des instances pathologiques pour des algorithmes d’approximation vus en cours. On combinera deux approches :

- la réflexion issue de l’analyse de l’algorithme et de ses limites ;
- l’exploration empirique basée sur de la génération aléatoire.

C’est en combinant ces deux approches que l’on maximise les chances de trouver la pire instance pathologique.

1 Couverture gloutonne

Le professeur Sourire propose l’heuristique suivante pour résoudre le problème de la couverture de sommets. On choisit de façon répétée un sommet de plus haut degré, et on supprime toutes ses arêtes incidentes. On s’arrête lorsqu’il n’y a plus d’arêtes. On souhaite construire une instance dont le facteur d’approximation soit le plus large possible.

Pour cela, on commence par créer un graphe quelconque qui facilitera l’implémentation de l’heuristique en fournissant la structure sur laquelle on travaille :

```
import networkx as nx
G = nx.Graph()
G.add_nodes_from([1, 2, 3, 4])
print(G.nodes())
G.add_edges_from([(1, 2), (2, 3), (1, 3), (3, 4)])
print(G.edges())
print(G[3])
```

On va maintenant implémenter deux méthodes :

- `verifier_couverture` qui retourne vrai si une couverture donnée est valide pour un graphe donné ;
- `heuristique_Sourire` qui génère une couverture pour un graphe.

Vous aurez peut-être besoin de la méthode `copy` pour copier le graphe.

On va désormais construire une instance pour laquelle l’heuristique du professeur Sourire est arbitrairement mauvaise. L’instance aura une structure de graphe biparti avec n sommets à gauche. Ensuite, pour tout $2 \leq i \leq n$: on rajoute $k = \lfloor n/i \rfloor$ sommets à droite de degré i , tous connectés aux sommets de gauche qui ont le plus petit degré.

On peut vérifier que l’ensemble des sommets de gauche constitue une couverture et que l’heuristique retourne une couverture de taille supérieure avec un ratio qui augmente avec n .

Pour quelle valeur de n obtient-on un facteur qui dépasse 2 ?

2 Tournée-VC-Approchée

L’approche précédente consiste à concevoir spécifiquement une instance qui exploite une faille de l’heuristique. Une autre approche consiste à générer aléatoirement des instances pour étudier le comportement.

Nous allons maintenant générer des graphes complets pondérés pour le problème du voyageur de commerce afin de déterminer s’il existe une instance dont le facteur d’approximation dépasse le $3/2$ qui a été trouvé en TD.

Il faudra calculer l'optimal pour évaluer la qualité de ce que retourne l'algorithme d'approximation. Cette recherche ne sera donc valable que pour des petites instances (4 à 8 sommets).

On va préparer la recherche en codant 3 parties : la construction d'un graphe ; le calcul de la tournée optimale ; l'implémentation de l'algorithme d'approximation.

Pour la construction du graphe, on utilisera la méthode suivante : on part d'un graphe complet avec une pondération unitaire sur chaque sommet. On sélectionne alors une arête au hasard pour augmenter sa pondération d'une unité si cela ne viole pas les inégalités triangulaires.

Le code suivant génère un graphe complet avec une pondération unitaire :

```
G = nx.complete_graph(n)
for e in G.edges():
    G[e[0]][e[1]]["weight"] = 1
```

Vous choisir une arête aléatoire, vous pourrez utiliser l'approche suivante :

```
edge = random.sample(list(G.edges()), 1)[0]
```

On peut désormais calculer la tournée optimale. Il faudra d'abord écrire le code d'une fonction pour évaluer le coût d'une tournée. Ensuite, on évaluera toutes les tournées et on choisira la meilleure. Pour générer toutes les tournées, on peut s'appuyer sur la bibliothèque `itertools` :

```
import itertools
itertools.permutations(G)
```

Il ne reste plus qu'à implémenter `TOURNÉE-VC-APPROCHÉE`.

On peut commencer la recherche empirique. Il faut tâtonner en gérant les paramètres numériques suivant :

- la taille du graphe ;
- le nombre de graphes générés ;
- la quantité de poids rajoutée lors de la génération aléatoire.

Quelle est l'instance la plus petite qui dépasse $3/2$?

3 Couverture-Ensemble-Glouton

On peut procéder de même pour le problème de la couverture d'ensemble afin de trouver une instance qui atteint un facteur d'approximation supérieur d'au moins 3 pour `COUVERTURE-ENSEMBLE-GLOUTON`.