**INF8775 – Analyse et conception d’algorithmes**

TP3 – Automne 2020

|  |  |
| --- | --- |
| **Nom, prénom, matricule des membres** | Laferrière, Thomas, 1905759 |
| **Note finale / 25** | **0** |

# Informations sur la correction

* Répondez directement dans le document DOCX.
* La correction se fait sur ce même rapport.
* Vous devez faire une remise électronique sur Moodle avant le 20 Avril à 23h59 en suivant les instructions suivantes :
  + Vos fichiers doivent être remis dans une archive zip à la racine de laquelle on retrouve :
    - Le rapport au format DOCX.
    - Un pdf contenant votre analyse asymptotique
    - Un script nommé *tp.sh* servant à exécuter les différents algorithmes du TP. L’interface du script est décrite à la fin du rapport.
    - Le code source et les exécutables.
    - Si le langage que vous utilisez nécessite une phase de compilation, veuillez joindre un Makefile afin que nous puissions le compiler en cas de problème avec vos exécutables. Si nous ne sommes pas en mesure de tester votre code, vous perdrez des points de respect d’interface et de qualité de code !
* Vous avez le choix du langage de programmation utilisé mais vous devrez utiliser les mêmes langage, compilateur et ordinateur pour toutes vos implantations. Le code et les exécutables soumis devront être compatibles avec les ordinateurs de la salle L-4714.
* Si vous utilisez des extraits de codes (programmes) trouvés sur Internet, vous devez en mentionner la source, sinon vous serez sanctionnés pour plagiat.

# Q1 – Description de votre algorithme

*Décrivez en quelques phrases votre algorithme. Soyez clair et concis. Donnez les noms des patrons de conception utilisés.*

|  |  |
| --- | --- |
| **0** | / 4 pt |

Mon algorithme se divise en deux parties : le départ et l’amélioration continue. Le départ est un algorithme glouton suivi d’une correction approximative. La partie gloutonne consiste à placer les arètes les moins énergétiques en touchant aux nœuds les plus connectés. Les nœuds qui sont restés vides sont ensuite remplis par n’importe quel atome restant afin d’avoir une solution valide.

L’amélioration continue effectue une recherche heuristique d’un voisinage aléatoire composé d’échanges de noeuds. L’heuristique utilisée est la somme de l’énergie de toutes les arètes connectées à un nœud. Il y a une composante taboue consistant à éviter de faire l’échange effectué au dernier tour afin d’éviter de tomber dans un minimum local, même si la composante aléatoire rend la probabilité en pratique que ça arrive très mince.

# Q2 – Présentation

*Sous forme de pseudo-code et incluant une analyse de complexité théorique des principales fonctions. Si vous préférez écrire vos équations en Latex, vous pouvez ajouter un pdf à la remise avec la réponse à cette question et le mentionner ici. Pas besoin de faire une analyse empirique de la complexité.*

|  |  |
| --- | --- |
| **0** | / 6 pt |

Voir pdf.

# Q3 – Justification de l’originalité de vote algorithme

*La conception de votre algorithme sera jugée avec les critères suivants :*

* *Lien avec le contenu du cours*
* *Originalité*

*Si vous vous êtes inspirés d’un algorithme existant, citez sa source.*

|  |  |
| --- | --- |
| **0** | / 4 pt |

Cet algorithme est original, car il combine deux patrons que nous avons vu en cours. L’algorithme de départ est un algorithme glouton approximatif avec une correction. L’algorithme d’amélioration consiste simplement en un algorithme numérique probabiliste combiné à un algorithme de recherche heuristique avec tabou pour éviter tomber dans un cycle de 2 échanges idempotent. En utilisant une composante probabiliste, on explore l’espace de manière variée et aléatoire, nous permettant de profiter du parallélisme accessible. De plus, après assez temps sans amélioration, on pourrait avoir une bonne confiance (0.9999...) que nous avons trouvé la réponse optimale.

# Q4 – Votre algorithme est-il assuré de trouver la solution optimale ?

*Répondez simplement “oui” ou “non”. Aucune justification requise.*

|  |  |
| --- | --- |
| **0** | / 1 pt |

Non.

# Autres critères de correction

## Qualité de l’algorithme

*Les points de cette question sont répartis comme suit :  
Nous allons exécuter votre code sur 3 exemplaires de notre choix. Pour chaque exemplaire la sortie de votre code sera envoyée vers le script de vérification, et nous classerons sur chaque exemplaire les différents binômes.  
Il y a 2 points par exemplaire. Le premier s’obtient en dépassant une baseline obtenue avec un algorithme basique. Le deuxième dépend de votre classement. Si votre algorithme donne une solution valide mais est classé dans le dernier quart vous aurez 0.25pt, si il est classé dans le 3eme quart vous aurez 0.5pt, si il est classé dans le 2eme quart, vous aurez 0.75pt. Enfin, si il est classé dans le meilleur quart, vous obtiendrez 1pt.*

*Ainsi si votre algorithme retourne des solutions valides dépassant la baseline, alors vous avez au moins 3,75 pts, indépendamment du classement.*

|  |  |
| --- | --- |
|  | / 6 pt |

## Respect de l’interface tp.sh

|  |  |
| --- | --- |
|  | / 1 pt |

## 

## Qualité du code

|  |  |
| --- | --- |
|  | / 2 pt |

## Présentation générale (concision, qualité du français, etc.)

|  |  |
| --- | --- |
|  | / 1 pt |

## Pénalités

|  |
| --- |
| 0 |

* Retard : -15 % / journée de retard, arrondi vers le haut. Les TPs ne sont plus acceptés après 3 jours.
* Autres : Le correcteur peut attribuer d’autres pénalités (par exemple si les exécutables sont manquants, etc.)