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

Rapport TP2 – Automne 2022

|  |  |
| --- | --- |
| **Nom, prénom, matricule des membres** | NOM, Prénom, 1234567  NOM, Prénom, 1234567 |
| **Note finale / 30** | 0 |

# Informations techniques

* Répondez directement dans ce document DOCX. Veuillez ne pas inclure le texte en italique servant de directive.
* La correction se fait sur ce même rapport.
* Vous devez faire une remise électronique sur Moodle avant le 15 Novembre à 23h59 en suivant les instructions suivantes :
  + Vos fichiers doivent être remis dans une archive zip à la racine de laquelle on retrouve :
    - Ce rapport au format DOCX.
    - 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.

Mise en situation

Ce travail pratique se répartit sur deux séances de laboratoire et porte sur l'analyse et la conception d'algorithmes développés suivant différents patrons de conception afin de résoudre une version simplifiée d'un problème réaliste d'optimisation.

La chaine de restauration rapide W&A songe à élargir sa zone de couverture dans plusieurs villes canadiennes suite à l'engouement général pour ses burgers au “poulet de grain élevé sans antibiotiques et nourri de manière végétarienne”. Une étude de marché lui a permis d'identifier pour chaque ville les sites potentiels où il est possible d'installer de nouveaux restaurants, ainsi que les revenus prévisionnels et la consommation de poulet attendue chaque jour, pour chaque site. Tous les restaurants d'une ville seront reliés à un seul fournisseur, qui livrera la totalité du poulet utilisé dans ses burgers. Afin de pouvoir fournir tous les restaurants de la ville, ce fournisseur ne peut pas livrer une quantité de poulet qui dépasse celle qu'il obtient des producteurs à chaque jour (sa capacité). Tous les fournisseurs sont déjà actifs et chacun d'eux a une quantité de poulet livrable fixe et limitée. Votre rôle est de trouver une solution pour chaque ville qui permet de maximiser la somme des revenus prévus tout en respectant la quantité de poulet livrable par chaque fournisseur. Comment faire ?

# Implantation

Trois algorithmes seront implantés, mettant en pratique les patrons de conception: glouton, programmation dynamique, heuristique d'amélioration locale et probabiliste.

## Algorithme glouton probabiliste

L'algorithme glouton fait son choix d'emplacement en fonction de la rentabilité. Pour chaque emplacement *i* vous calculez sa rentabilité *Ri* comme étant le revenu divisé par la quantité de poulet :

*Ri = ri /qi.*

Nous ajoutons un ingrédient probabiliste en randomisant l'algorithme : à chaque itération, plutôt que d'ajouter l'emplacement qui a la rentabilité maximum, on choisit au hasard un emplacement proportionnellement à sa rentabilité.

La probabilité de choisir l'emplacement *i* est *pi = Ri / ∑j=1..n Rj .*

Un exemple d'implémentation est disponible à la page suivante : <http://en.wikipedia.org/wiki/Fitness_proportionate_selection>

L'algorithme est lancé à dix reprises et la meilleure solution est retournée.

## Algorithme de programmation dynamique

L'algorithme de programmation dynamique remplit un tableau de revenus optimaux en fonction des sites permis et de la quantité livrable du fournisseur : *D[i,j] = max(ri + D[i-1, j-qi], D[i-1, j])* représente le meilleur revenu possible pour un fournisseur ayant la quantité *j* en utilisant des sites parmi les *i* premiers.

Cet algorithme peut être assez gourmand en espace mémoire.

## Heuristique d’amélioration locale

L'algorithme de type amélioration locale démarre sa recherche avec une première solution connue qui est celle obtenue par l'algorithme glouton. Ensuite, pour améliorer cette solution, il est possible de réaliser un certain nombre d'améliorations locales.

L'heuristique d'amélioration locale entre dans la grande famille des algorithme de recherche locale. Ces algorithmes utilisent deux concepts : le voisinage et la fonction objectif. Le voisinage est la différence entre deux solutions, dites voisines. Pour cet algorithme, il s'agira d'enlever un ou deux sites et d'en ajouter un ou deux autres. Évidemment, vos nouvelles solutions doivent demeurer valides. La fonction objectif est le revenu généré par l'ensemble des sites choisis. Par conséquent, à chacune des itérations de l'algorithme, vous devez choisir, parmi tous les voisins, celui qui améliore le plus la fonction objectif. Vous devez évidemment tenir compte de la validité de la nouvelle solution. Cela signifie que si un échange implique un excès de la capacité du fournisseur, il ne peut pas être choisi.

Le critère d'arrêt sera l'optimum local : lorsqu'aucune solution voisine améliorant la fonction objectif est trouvée, l'algorithme s'arrête.

# Jeu de données

Vous trouverez sur le site Moodle du cours tous les exemplaires (villes) du problème à résoudre.

La structure des fichiers d'exemplaires est :

Sur la première ligne: *Nombre d'emplacements*

Une ligne pour chaque emplacement: *i* <espace> *ri*<espace> *qi*

Sur la dernière ligne: *Capacité du fournisseur (quantité livrable)*

# Présentation des résultats

|  |  |
| --- | --- |
|  | / 1,5pts |

## Tableau des résultats

*Exécutez chacun des trois algorithmes en notant leur temps d'exécution et la hauteur maximale de votre tour, mais ne rapportez dans un tableau que la moyenne de chacune des séries de dix exemplaires.*

*Pensez à indiquer l'unité de temps utilisée.*

# Analyse et discussion

### Faites une analyse asymptotique théorique du temps de calcul pour chaque algorithme.

|  |  |
| --- | --- |
|  | / 7,5 pt |

*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. Justifiez votre analyse. Veillez à indiquer la complexité de chaque étape clé, même de celles qui peuvent devenir négligeables face à d’autres étapes plus complexes. Nous devons voir que vous les avez bien prises en compte.*

### Servez-vous de vos temps d'exécution pour confirmer et/ou préciser l'analyse asymptotique théorique de vos algorithmes avec la méthode hybride de votre choix.

|  |  |
| --- | --- |
|  | / 7,5 pt |

*La méthode peut varier d'un algorithme à l'autre. Justifiez les choix ici et montrez vos graphiques. Attention, vous devrez vous poser des questions pour l’algorithme d’amélioration locale.*

### Discutez des trois algorithmes en fonction de la qualité respective des solutions obtenues, de la consommation de ressources (temps de calcul, espace mémoire) et de la difficulté d'implantation.

### Indiquez sous quelles conditions vous utiliseriez chaque algorithme.

|  |  |
| --- | --- |
|  | / 6,5 pt |

## 

# Autres critères de correction

## Respect de l’interface tp.sh

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

## Utilisation :

tp.sh –a [glouton|progdyn|local] -e path\_vers\_exemplaire

Arguments optionnels :

-p affiche la solution (emplacements i choisis) sur une ligne, avec chaque emplacement séparé par un espace

Exemple :

1 4 12 20

-t affiche le temps d’exécution en ms, sans unité ni texte superflu

Important : l’option -e doit accepter des fichiers avec des paths absolus.

## Qualité du code

|  |  |
| --- | --- |
|  | / 5 pt |

* + - * 1. Validité des solutions
        2. Qualité de l'implémentation

Présence de commentaires

## Présentation générale

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

* Concision
* Qualité du français

## Pénalité retard

|  |
| --- |
| 0 |

* -15% de la note / journée de retard, arrondi vers le haut. Les TPs ne sont plus acceptés après 3 jours.