INF4705 — Analyse et conception d'algorithmes
TP 2

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.

1 Problématique

La chaine de restauration rapide $\mathcal{W}\&\mathcal{A}$ 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 ?

2 Implantation

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

2.1 Algorithme vorace probabiliste

L'algorithme vorace fait son choix glouton d'emplacement en fonction de la rentabilité. Pour chaque emplacement i vous calculez sa rentabilité $\overline{r}_i$ comme étant le revenu divisé par la quantité de poulet : $\overline{r}_i = r_i/q_i$.

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 $p_i = \overline{r}_i / \sum_{j=1}^{N} \overline{r}_j$. 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.

2.2 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 :

$$R[i, j] = \max(r_i + R[i-1, j-q_i], R[i-1, j])$$

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.

2.3 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 vorace. 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.

3 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> r_i <espace> q_i
Sur la dernière ligne : Capacité du fournisseur (quantité livrable)

4 Résultats

Exécutez chacun des trois algorithmes en notant leur temps d'exécution et le revenu total de la solution trouvée pour chaque exemplaire mais ne rapportez dans un tableau que la moyenne de chaque série de dix exemplaires.

5 Analyse et discussion

1. Faites une analyse asymptotique du temps de calcul pour chaque algorithme (ou au moins pour certaines parties de l'algorithme).
2. 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 (cette méthode peut varier d'un algorithme à l'autre). Justifiez ces choix.
3. 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.
4. Indiquez sous quelles conditions vous utiliseriez l'un de ces algorithmes plutôt que les deux autres.

6 Remise

Avant votre cinquième séance de laboratoire, vous devez faire une remise électronique en suivant les instructions sur le site du cours. Votre répertoire devra comprendre les éléments suivants :

1. un rapport comprenant :
   — une brève description du sujet et des objectifs de ce travail (svp pas de redite de l'énoncé),
   — la description du jeu de données,
   — les résultats expérimentaux,
   — l'analyse et discussion
2. les trois exécutables nommés : vorace, dynamique et local.
3. un fichier *ReadMe.txt* indiquant les arguments à entrer en ligne de commande lors de l'exécution de vos programmes.

Les exécutables doivent respecter le standard suivant :

1. Chaque exécutable doit avoir un des nom suivant, selon sa fonction.
2. Chaque exécutable doit prendre les paramètres suivant : -f chemin vers l'exemplaire et -p pour imprimer les emplacements choisis.
3. Chaque exécutable, lorsqu'il est exécuté sans le paramètre -p, doit uniquement afficher son temps d'exécution. Vous devrez juger de la précision nécessaire pour vos tests.

Si le standard n'est pas respecté, des pénalités s'ensuivront. Ce format est cependant pratique, car il vous permettra de standardiser facilement vos tests, avec un script par exemple. Par ailleurs, ne remettez pas les fichier de données (exemplaires).

7 Barème de correction

1 pts : exposé du travail pratique
2 pts : présentation des résultats
5 pts : analyse et discussion
3 pts : les programmes (corrects,structurés,commentés,...)
2 pts : présentation générale et qualité du français