Rapport du projet de programmation linéaire

Maxence Ahlouche Maxime Arthaud Korantin Auguste
Martin Carton Thomas Forgione Thomas Wagner

21 octobre 2013

Table des matières

1 Présentation de l'équipe

Cette équipe a été menée par Maxence Ahlouche, assisté de son Responsable Qualité Thomas Wagner. Les autres membres de l'équipe sont Martin Carton, Thomas Forgione, Maxime Arthaud, et Korantin Auguste.

2 Problème du sac à dos

2.1 Résolution exacte

Nous avons implémenté un algorithme de programmation dynamique, qui permet de résoudre le problème du sac à dos. Toutefois, il fonctionne uniquement si les poids des objets sont des entiers.

Sa complexité en temps est en $O(nW)$ et celle en mémoire en $O(W)$, avec n le nombre d'objets et W le poids maximum du sac.

Nous l'avons testé sur plusieurs instances du problème (jusqu'à X objets et un poids maximal de X), et l'algorithme s'exécute toujours en moins d'une seconde.

2.2 Résolution approchée

todo : parler du glouton, comparer les critères de tri

3 Annexe

Listings

Listing 1 – Codes relatifs au problème du sac à dos

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

def sacados(objets, masse_max):
    """
    Résoud le problème du sac à dos avec de la programmation dynamique.
    Fonctionne seulement avec des valeurs entières.

    >>> objets = ((2,3),(3,4),(4,5),(5,6))
    >>> sacados(objets, 5)
    7
```

```
    """
    assert isinstance(masse_max, int) and all(isinstance(x[0], int) for x in
        objets)

    current_line = [0 for i in range(masse_max+1)]
    prev_line = current_line[:]

    for i in range(0,len(objets)):
        masse_objet, prix = objets[i]
        for masse in range(masse_max + 1):
            if masse_objet <= masse:
                current_line[masse] = max(prev_line[masse],
                    prev_line[masse-masse_objet] + prix)
            else:
                current_line[masse] = prev_line[masse]

        prev_line = current_line[:]

    return current_line[masse_max]

def best_ratio(x): return x[1]/x[0]
def less_mass(x):  return -x[0]
def best_price(x): return x[1]

def greedy(objects, max_mass, key):
    """
        Algorithme approché du glouton.
        Nécessite de trier les objets selon un critère `key`.
        Par exemple
            greedy(obj, max_mass, less_mass)
        choisit les objets en commençant par les moins lourds.
    """
    cost, mass = 0, 0
    objects = sorted(objects, key=key, reverse=True)

    for o in objects:
        if o[0] + mass <= max_mass:
            mass += o[0]
            cost += o[1]

            if mass == max_mass:
                break

    return cost

def read_testfile(path):
    """
        Lit un fichier généré par le générateur trouvé ici:
            http://www.diku.dk/~pisinger/codes.html
        Retourne une liste de couples (masse, valeur) considérée comme bon
        exemple.
    """
    with open(path, 'r') as f:
        objects = []
        line = f.readline()
        nb_objs = int(line)
        for i in range(0, nb_objs):
            line = f.readline()
            dummy, a, b = map(int, line.split())
            objects.append((b, a))
        return objects
```

```
if __name__ == '__main__':
    import doctest
    doctest.testmod()
```