Mini-projet d'algorithmique avancée

Kévin VYTHELINGUM, Jean-Michel NOKAYA

30 décembre 2013

Table des matières

1 Introduction **2**

2 Préliminaires **2**

3 Méthodes des essais successifs **2**
3.1 Analyse . . . 2
3.2 Condition d'élagage . . . 3
3.3 Algorithme . . . 3
3.3.1 Sans élagage . . . 3
3.3.2 Avec élagage . . . 4
3.4 Complexité temporelle . . . 4
3.5 Expérimentations . . . 4

4 Programmation dynamique **4**
4.1 Formule de récurrence . . . 4
4.2 Structure tabulaire . . . 4
4.3 Algorithme . . . 4
4.4 Complexité temporelle . . . 4
4.5 Complexité spatiale . . . 4

5 Question complémentaires **4**

6 Conclusion **4**

1 Introduction

2 Préliminaires

1.

$$\begin{aligned} d_{max} &\Rightarrow c_{min} \\ \sum_{i=1}^{k} d_{max} &\Rightarrow \sum_{i=1}^{n} c_{min} \\ D &\Rightarrow C_{opt} \end{aligned}$$

2. complexité $= k^n$ (On peut faire un arbre pour le démontrer)

3 Méthodes des essais successifs

3.1 Analyse

Solution : un candidat est un vecteur de taille n où chaque coefficient est une durée choisie parmi l'ensemble {1,...,k} (à chaque tâche on associe une durée). On choisit d'enregistrer les choix réalisés dans un tableau T de taille n.

S_i : l'ensemble des durées possibles de 1 à k

$satisfaisant(x_i) = \sum_{j=1}^{i} x_j \leq D$
(la somme partielle des durées choisies est inférieure à la durée maximale autorisée)

$enregistrer(x_i) = T[i] \leftarrow x_i$

$soltrouvee$: $i = n$

$defaire(x_i) = T[i] \leftarrow 0$

Pour simplifier les vérifications au niveau de satisfaisant et des conditions d'élagage, on utilisera les variables entières *cout* et *duree* initialisée à 0, qui représenteront le cout courant et la durée courante duent à nos choix de durées. Elles seront mises à jour dans *enregistrer* et dans *défaire*. En effet, en notant *CD* le tableau à deux dimensions ayant les coûts en ligne et les durées en colonne, on effectura dans *enregistrer* :

$$\begin{aligned} \text{coût} &\leftarrow \text{coût} + CD[i][x_i] \\ \text{durée} &\leftarrow \text{durée} + x_i \end{aligned}$$

Ensuite on effectura dans *défaire* :

$$\begin{aligned} \text{coût} &\leftarrow \text{coût} - CD[i][x_i] \\ \text{durée} &\leftarrow \text{durée} - x_i \end{aligned}$$

3.2 Condition d'élagage

3.3 Algorithme

3.3.1 Sans élagage

Cet algorithme repose sur le modèle de programmation de recherche d'une solution optimale par essais successifs. Le principe est d'essayer toutes les combinaisons possibles tant qu'elles sont satisfaisantes, c'est-à-dire dans notre cas que la somme partielle des durées choisies est inférieure à la durée maximale autorisée D. Pour obtenir la solution optimale, on vérifie que le coût d'une solution obtenue est strictement inférieur au coût de la meilleure solution trouvée, auquel cas on l'affiche. La dernière solution affichée sera donc la solution optimale.

L'appel de l'algorithme consiste à initialiser le coût optimal au coût maximal qu'il est possible d'obtenir, puis à appeler la procédure au rang 1, c'est-à-dire commencer par choisir la durée de la première tâche T_1.

procédure *ordonnancement_simple* (ent i) ;
var ent n, k, x_i, D ;
var ent cout, duree ;
début
    $cout \leftarrow 0$;
    $duree \leftarrow 0$;
    pour x_i **de** 1 **à** k **faire**
        si $duree \leq D$ **alors**
            $T[i] \leftarrow x_i$;
            $cout \leftarrow cout + cd[i][x_i]$;
            $duree \leftarrow duree + x_i$;
            si i = n **alors**
                si $cout < cout\_opt$ **alors**
                    $affiche(T)$;
                fsi ;
            sinon
                $ordonnancement\_simple(i + 1)$;
            fsi ;
            $T[i] \leftarrow 0$;
            $cout \leftarrow cout - cd[i][x_i]$;
            $duree \leftarrow duree - x_i$;
        fsi ;
    fait ;
fin ;

Appel : $cout\_opt \leftarrow c_{max}$; $ordonnancement\_simple(1)$;

3.3.2 Avec élagage

3.4 Complexité temporelle

3.5 Expérimentations

4 Programmation dynamique

4.1 Formule de récurrence

4.2 Structure tabulaire

4.3 Algorithme

4.4 Complexité temporelle

4.5 Complexité spatiale

5 Question complémentaires

1.
2.
3.
4.
5.

6 Conclusion