

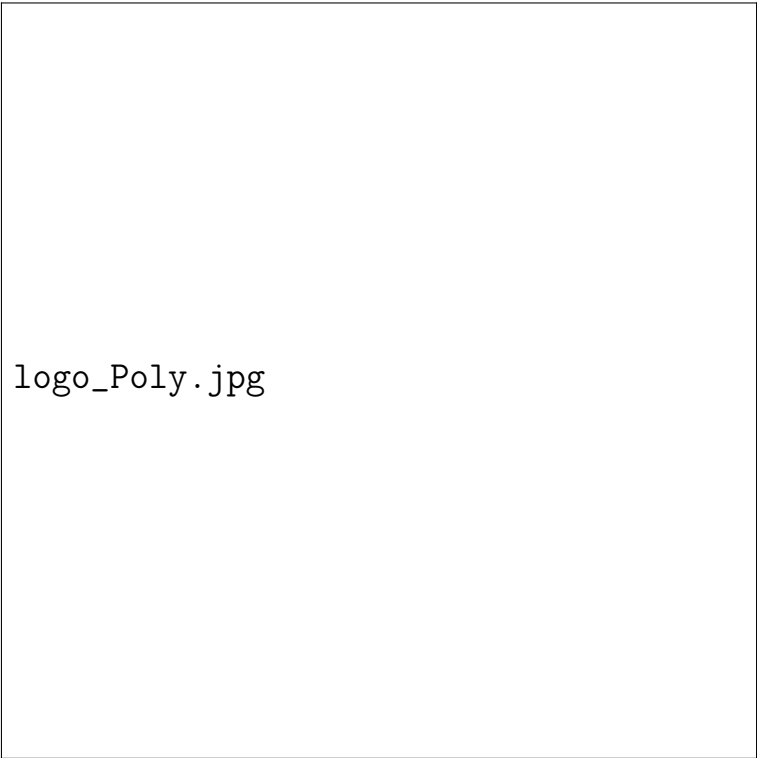
Analyse et conception d'algorithmes

INF4705

L4714

Jérémy CUGMAS (1814477) & Juliette TIBAYRENC (1800292)

3 nov. 2015



logo_Poly.jpg

Table des matières

0. Introduction	2
1. Revue de la théorie	3
2. Protocole expérimental	4

0. Introduction

Ce deuxième TP se penche sur le problème de mise en boîte, plus particulièrement dans le cas où toutes les boîtes sont identiques et qu'il faut placer dans chacune d'elles un nombre précis d'items avec des volumes variables sans dépasser leur capacité. Ce problème relève de l'optimisation combinatoire et son objectif principal est d'utiliser de manière optimale le volume des boîtes pour placer les items sans dépasser le volume maximum autorisé. Plus historiquement, ce problème fait partie des 21 problèmes NP-complet, c'est-à-dire qu'il appartient à la classe des problèmes qui peuvent être résolus en temps polynomial sur une machine non déterministe. De plus, ce problème est utilisé dans de nombreuses situations actuelles comme le chargement des bagages dans un avion ou encore dans le découpe des matériaux pour diminuer au maximum les pertes. Afin de le résoudre, nous allons implanter trois algorithmes : vorace, programmation dynamique et recuit simulé afin de comparer leur efficacité et la qualité des solutions que propose chacun de ces algorithmes.

1. Revue de la théorie

Comme énoncé précédemment, ce TP a pour but d'élaborer un algorithme vorace, de programmation dynamique et de recuit simulé. Le premier d'entre eux repose sur une succession de choix qui semblent à chaque fois être le plus optimal mais malheureusement sans la possibilité de revenir en arrière. Les algorithmes de type « vorace » ou « glouton » ont la particularité d'être simple à concevoir, efficace et assez simple à implémenter mais ne représentent souvent pas la méthode la plus optimale. Le plus souvent, l'algorithme vorace dans le cas du problème de mise en boîtes se divise en deux étapes majeures : le tri des items et le placement de ceux-ci dans les boîtes qui ont respectivement les complexités, pour un nombre n d'items, $\Theta(n \log(n))$ et $\Theta(n)$ donc d'après la règle du maximum la complexité globale est de $\Theta(n \log(n))$.

Ensuite nous avons l'algorithme de programmation dynamique, celui-ci se base sur le principe d'optimalité de Bellman qui affirme qu'une solution optimale s'obtient en résolvant des sous-problèmes de façon optimale. En résumé, cela signifie que la solution optimale d'un problème s'obtient par la combinaison des solutions optimales des sous-problèmes. A contrario de l'algorithme diviser pour régner, la programmation dynamique procède en « bottom-up », c'est-à-dire qu'on débute par la résolution des sous-problèmes les plus petits puis on traite progressivement les plus gros. Avec l'algorithme dynamique, pour faciliter la vision du problème, on définit un tableau contenant la solution de chacun des sous-exemplaires de l'exemplaire originel dans un certain ordre. Dans le cas du problème, la complexité se mesure par le remplissage du tableau avec les n éléments et le rebrousser ensuite pour trouver notre solution. Cela nous donne une complexité globale de $\Theta(n^2)$.

Le dernier algorithme est le recuit simulé qui est une méthode type métaheuristique utilisée à la base en thermodynamique. Il repose sur le choix aléatoire d'une solution donnée pour ensuite la modifier afin d'en obtenir une seconde, soit celle-ci améliore le critère d'optimisation soit elle le dégrade. On parle alors de voisinage. Dans le cas de la mise en boîtes, la solution voisine est obtenue en choisissant uniformément au hasard un item non placé et on l'insère dans une boîte choisie au hasard. Si cela fait déborder la boîte, on choisit un item au hasard dans cette boîte pour respecter la capacité de celle-ci.

2. Protocole expérimental