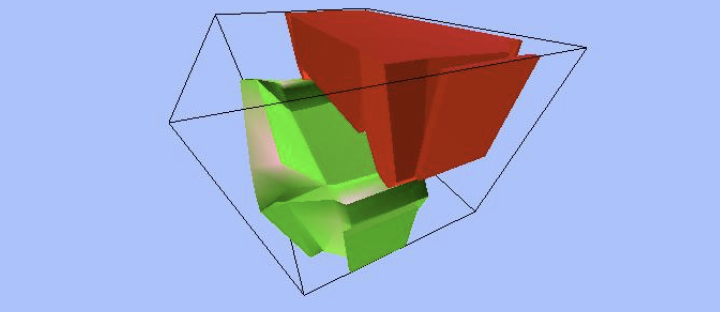
3D Knapsack Problem



Group 10

Annelot Pruyn, Benjamin Rodrigues de Miranda, Costanza Siani, Olive Kirk, Shir-lee Kimelman, Tanguy Legrand

Block 1.3

University of Maastricht

Department of data science and knowledge engineering

3D knapsack problem

Block 1.3

Group 10

Maastricht, 25.01.2018

Carlo Galuzzi

Preface

In block 1.3, our goal was to create a user-friendly interface that can be used to solve a 3D knapsack problems. Our project is useful as cargo companies could use it to maximize the density of a container’s value, making cargo transfer more efficent.

/\* The knapsack Problems are mainly a “Combinatorial Optimization Problem”, which consists in looking for a way to maximize or minimize some “quantity” while satisfying specific constrains. In this task, we are talking about maximizing the obtained profit without exceeding the knapsack capacity, which is a very special case of integer Linear Programming Problem.

Therefore, the problem can also be defined as “a greedy method in which knapsack is nothing but a bag which consists of n objects each object associated with weight and profit”( **S. P. Sajjan, Ravi kumar Roogi, Vijay kumar Badiger, Sharanu Amaragatti “ A new approach to solve Knapsack Problem” )**. \*/

Chapter 1: Introduction

1.1. Problem analysis

The knapsack Problems are mainly a “Combinatorial Optimization Problem”, which consists in looking for a way to maximize or minimize some “quantity” while satisfying specific constrains. In this task, we are talking about maximizing the obtained profit without exceeding the knapsack capacity, which is a very special case of integer Linear Programming Problem.

Therefore, the problem can also be defined as “a greedy method in which knapsack is nothing but a bag which consists of n objects each object associated with weight and profit” (**S. P. Sajjan, Ravi kumar Roogi, Vijay kumar Badiger, Sharanu Amaragatti “ A new approach to solve Knapsack Problem” )**.

1.2. Objectives (performance criteria)

In the knapsack problem, every object has a weight w(i) and a value c(i), W is the maximum weight of the knapsack and the possibility that an object could or could not be insert in, is expressed with the integer variable x(i).

The goal is to find a Z using an equation like:

max Z =

by keeping in mind, the following constrain:

In our case however, we’re not working with weight but rather with volume, given that we have to fit certain parcels into the container the volume is our restriction instead of the weight.

so:

1.2.1 volume

Chapter 2 Possible solutions:

2.1 Knapsack problems:

According to the type of the variable x(i):

- 0/1 Knapsack problem

x(i)={0,1} ∀i = 1,..,n

-> no repetitions

- bounded knapsack problem

x(i)≤ b(i) ∀i=1,…,n

-> tot repetitions

- unbounded Knapsack problem

x(i)∈ℵ ∀i=1,…,n

-> as many repetitions as wanted

The unbounded knapsack problem is the one we’ll need for this project

In order to solve this type of problem there is no polynomial-time algorithm. However, there is a [pseudo-polynomial time](https://en.wikipedia.org/wiki/Pseudo-polynomial_time) algorithm using [dynamic programming](https://en.wikipedia.org/wiki/Dynamic_programming), a technic based on the division of the big problem in smaller ones and on the use of optimal under structure .

Table of contents

Chapter 1: Introduction

* 1.1. Problem analysis
* 1.2. Objectives (performance criteria)
  + 1.2.1. Volume
  + 1.2.2. Value

Chapter 2: Possible solutions

* 2.1. Knapsack problems
* 2.2. Adaption to our problem (specifying multi dimensions unbounded)
* 2.3. Possible algorithms
  + 2.3.1. Backtracking
  + 2.3.2. Greedy
  + 2.3.3. Dynamic programming
* 2.4. Possible selection
  + 2.4.1. Random
  + 2.4.2. Ratio
* 2.5. Discussion

Chapter 3: Code model

Chapter 4: Simulation and Results

Chapter 5: Discussion and Conclusion