University Nice Côte d'Azur — Graphs
Master 1 — March-April 2019

Exercises : Knapsack Problem

To be returned for **April 16th 2019**.

The Simple Knapsack problem takes a set of integers $\mathcal{S} = \{w_1, \cdots, w_n\}$ and an integer b as inputs. The objective is to compute a subset $T \subseteq \{1, \cdots, n\}$ of items such that $\sum_{i \in T} w_i \leq b$ and $\sum_{i \in T} w_i$ is maximum. That is, we want to fill our knapsack without exceeding its capacity b and putting the maximum total weight in it.

1 Exact Algorithm via dynamic programming

Dynamic programming is a generic algorithmic method that consists in solving a problem by combining the solutions of sub-problems.

As an example, the Simple Knapsack Problem consists in computing an optimal solution for an instance $\mathcal{S} = \{w_1, \cdots, w_n\}$ and an integer b. Let $OPT(\mathcal{S}, b)$ denote such a solution. We will compute it using solutions for sub-problems with inputs $\mathcal{S}_i = \{w_1, \cdots, w_i\}$ and $b' \in \mathbb{N}$, for any $i \leq n$ and $b' < b$. That is, we will compute $OPT(\mathcal{S}, b)$ from all solutions $OPT(\mathcal{S}_i, b')$ for $i \leq n$ and $b' < b$.

Algorithm 1 Dynamic programming algorithm for Simple Knapsack

Require: A set of integers $\mathcal{S} = \{w_1, \cdots, w_n\}$ and $b \in \mathbb{N}$.
Ensure: A subset $OPT \subseteq \{1, \cdots, n\}$ of items such that $\sum_{i \in T} w_i \leq b$

For any $0 \leq i \leq n$ and any $0 \leq b' \leq b$, let $OPT[i, b'] = \emptyset$;
For any $0 \leq i \leq n$ and any $0 \leq b' \leq b$, let $opt\_cost[i, b'] = 0$;
for $i = 1$ to n **do**
  for $b' = 1$ to b **do**
    if $w_i \leq b'$ and $opt\_cost[i-1, b'-w_i] + w_i > opt\_cost[i-1, b']$ **then**
      $OPT[i, b'] = OPT[i-1, b'-w_i] \cup \{i\}$
      $opt\_cost[i, b'] = opt\_cost[i-1, b'-w_i] + w_i$
    else
      $OPT[i, b'] = OPT[i-1, b']$
      $opt\_cost[i, b'] = opt\_cost[i-1, b']$
return $OPT = OPT[n, b]$

Question 1 *Prove that Algorithm 1 has time-complexity $O(n \cdot b)$.*

Question 2 *Explain that we may assume that* $\max_i w_i \leq b$ *and* $b \leq \sum_i w_i$ *since, otherwise, the instance may be simplified.*

Prove that, if $\max_i w_i \leq b \leq \sum_i w_i$, *Algorithm 1 proceed in polynomial-time if* $\max_i w_i$ *is polynomial in n but exponential if* $\max_i w_i$ *is exponential in n.*

Actually, the Knapsack Problem is an example of *Weakly NP-hard* (roughly, it can be solved in polynomial-time if the weights are polynomial).

Question 3 *Prove by induction on i and b' that the solution $OPT = OPT[n, b]$ returned by Algorithm 1 is optimal.*

2 Approximation Algorithm and PTAS

Algorithm 2 Greedy algorithm for SIMPLE KNAPSACK

Require: A set of integers $\mathcal{S} = \{w_1, \cdots, w_n\}$ and $b \in \mathbb{N}$.

Ensure: A subset $T \subseteq \{1, \cdots, n\}$ of items such that $\sum_{i \in T} w_i \leq b$

$T = \emptyset$
$total\_weight = 0$
Sort $\mathcal{S}$. Let us assume that $w_1 \geq w_2 \geq \cdots \geq w_n$.
for $i = 1$ to n **do**
    if $total\_weight + w_i \leq b$ **then**
        Add i to T
        Add w_i to $total\_weight$
return T

Question 4 *What is the time-complexity of Algorithm 2 ?*

Question 5 *Prove that Algorithm 2 is a* 2*-approximation algorithm for the* SIMPLE KNAPSACK *problem.*

hint : let T be the computed solution and assume it is not optimal. Let $j \geq 1$ be the smallest integer such that $j+1$ is NOT in T. Show that $w_{j+1} \leq \sum_{i \leq j} w_i / j$ and that $\sum_{i \leq j} w_i \leq \sum_{i \in T} w_i \leq OPT < \sum_{i \leq j+1} w_i$ with OPT the value of an optimal solution.

A **polynomial-time approximation scheme (PTAS)** is an algorithm which takes an instance of an optimization problem and a parameter $\epsilon > 0$ and, in polynomial time in the size of the instance (not necessarily in ϵ), produces a solution that is within a factor $1 + \epsilon$ of being optimal.

That is, when ϵ tends to 0, the solution tends to an optimal one, while the complexity increases (generally, the complexity is of the form $O(n^{f(1/\epsilon)})$ for some function f).

Question 6 *Prove that Algorithm 3 has time-complexity $O(n^{\lceil 1/\epsilon \rceil + 1})$.*

hint : prove that there are $O(n^k)$ subsets of size at most k in a ground-set with n elements.

Question 7 *Prove that Algorithm 3 is a $(1+\epsilon)$-approximation algorithm for the* SIMPLE KNAPSACK *problem.*

hint : Consider an optimal solution M and let $X^* = \{i_1, \cdots, i_k\}$ be the k items with largest weight in M. Consider the iteration of Algorithm 3 when it considers X^*.

Actually, we can do better. Indeed, the KNAPSACK Problem admits a **fully polynomial-time approximation scheme (FPTAS)** algorithm, that is an algorithm that computes a solution that is within a factor $1 + \epsilon$ of being optimal **in time polynomial both in the size of the instance AND in** $1/\epsilon$.

Algorithm 3 PTAS for Simple Knapsack

Require: A set of integers $\mathcal{S} = \{w_1, \cdots, w_n\}$, $b \in \mathbb{N}$ and a real $\epsilon > 0$.

Ensure: A subset $T \subseteq \{1, \cdots, n\}$ of items such that $\displaystyle\sum_{i \in T} w_i \leq b$

$best = \emptyset$
$best\_cost = 0$
$k = \lceil 1/\epsilon \rceil$
for Any subset $X \subseteq \mathcal{S}$ of size k **do**
    Complete X using the Greedy Algorithm. That is :
    $T = X$
    $total\_weight = \sum_{i \in X} w_i$
    Sort $\mathcal{S} \setminus X$. Let us assume that $\mathcal{S} \setminus X = \{w_1, \cdots, w_{n-k}\}$ and $w_1 \geq w_2 \geq \cdots \geq w_{n-k}$.
    for $i = 1$ to $n - k$ **do**
        if $total\_weight + w_i \leq b$ **then**
            Add i to T
            Add w_i to $total\_weight$
    if $total\_weight > best\_cost$ **then**
        Replace $best$ by T
return T