

The background features a complex network graph with red nodes and black edges. The nodes are scattered across the frame, with some forming a path and others branching off. The background is filled with overlapping, colorful circles in shades of blue, yellow, red, and purple, creating a vibrant, abstract pattern.

# Lecture: Approximation Algorithms

Jannik Matuschke

TUM

November 5, 2018

# Dynamic Programming

## Example I: The Knapsack Problem

# The Knapsack Problem

- Input:** set of  $n$  items  $I$ , capacity  $B$ ,  
for each item  $i \in [n]$ : value  $v_i$ , size  $s_i$  (all integers)
- Task:** find  $S \subseteq I$  with  $\sum_{i \in S} s_i \leq B$ ,  
maximizing value  $\sum_{i \in S} v_i$

## A dynamic program

**Idea:** store all “good” subsets of  $\{1, \dots, j\}$  in  $A(j)$

## A dynamic program

**Idea:** store all “good” subsets of  $\{1, \dots, j\}$  in  $A(j)$

**Dominance:**  $X \succeq Y \iff s(X) \leq s(Y)$  and  $v(X) \geq v(Y)$

We don't need  $Y$  if we have  $X$  ...

# A dynamic program

**Idea:** store all “good” subsets of  $\{1, \dots, j\}$  in  $A(j)$

**Dominance:**  $X \succeq Y \iff s(X) \leq s(Y)$  and  $v(X) \geq v(Y)$

We don't need  $Y$  if we have  $X$  ...

## Algorithm 1 (DP for Knapsack)

- 1  $A(0) := \{\emptyset\}$
- 2 for  $j := 1$  to  $n$   
     $A(j) := A(j - 1)$   
    for each  $X \in A(j)$   
        if  $s(X) + s_j \leq B$  then  
            add  $X \cup \{j\}$  to  $A(j)$   
    while  $(\exists X, Y \in A(j)$  with  $X \succeq Y$ )  
        remove  $Y$  from  $A(j)$
- 3 return  $X \in A(n)$  maximizing  $v(X)$

## An approximation scheme

**Idea:** make  $V$  smaller by scaling all  $v_i$  down (and rounding)

## An approximation scheme

**Idea:** make  $V$  smaller by scaling all  $v_i$  down (and rounding)

Let's try to get a  $(1 - \varepsilon)$ -approximation for some  $\varepsilon > 0$ .



## An approximation scheme

**Idea:** make  $V$  smaller by scaling all  $v_i$  down (and rounding)

Let's try to get a  $(1 - \varepsilon)$ -approximation for some  $\varepsilon > 0$ .

### Algorithm 2 (FPTAS for Knapsack)

- 1  $M := \max\{v_i : i \in [n], s_i \leq B\}$ ,  $\mu := \frac{\varepsilon M}{n}$
- 2  $v'_i := \lfloor v_i / \mu \rfloor$  for all  $i \in [n]$
- 3 Solve instance with  $v'$  instead of  $v$ , using Algorithm 1.

## An approximation scheme

**Idea:** make  $V$  smaller by scaling all  $v_i$  down (and rounding)

Let's try to get a  $(1 - \varepsilon)$ -approximation for some  $\varepsilon > 0$ .

### Algorithm 2 (FPTAS for Knapsack)

- 1  $M := \max\{v_i : i \in [n], s_i \leq B\}$ ,  $\mu := \frac{\varepsilon M}{n}$
- 2  $v'_i := \lfloor v_i / \mu \rfloor$  for all  $i \in [n]$
- 3 Solve instance with  $v'$  instead of  $v$ , using Algorithm 1.

### Polynomial-time Approximation Scheme (PTAS):

$(1 - \varepsilon)$ -approximation for every  $\varepsilon > 0$

### Fully Polynomial-time Approximation Scheme (FPTAS):

$(1 - \varepsilon)$ -approximation for every  $\varepsilon > 0$ ,

running time polynomial in encoding and  $1/\varepsilon$