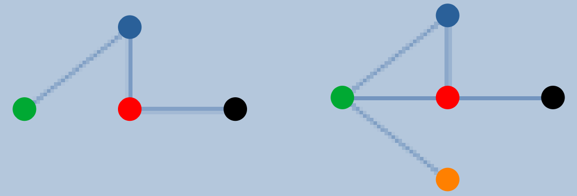
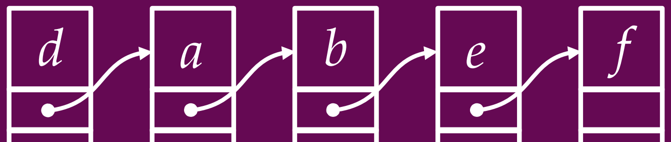
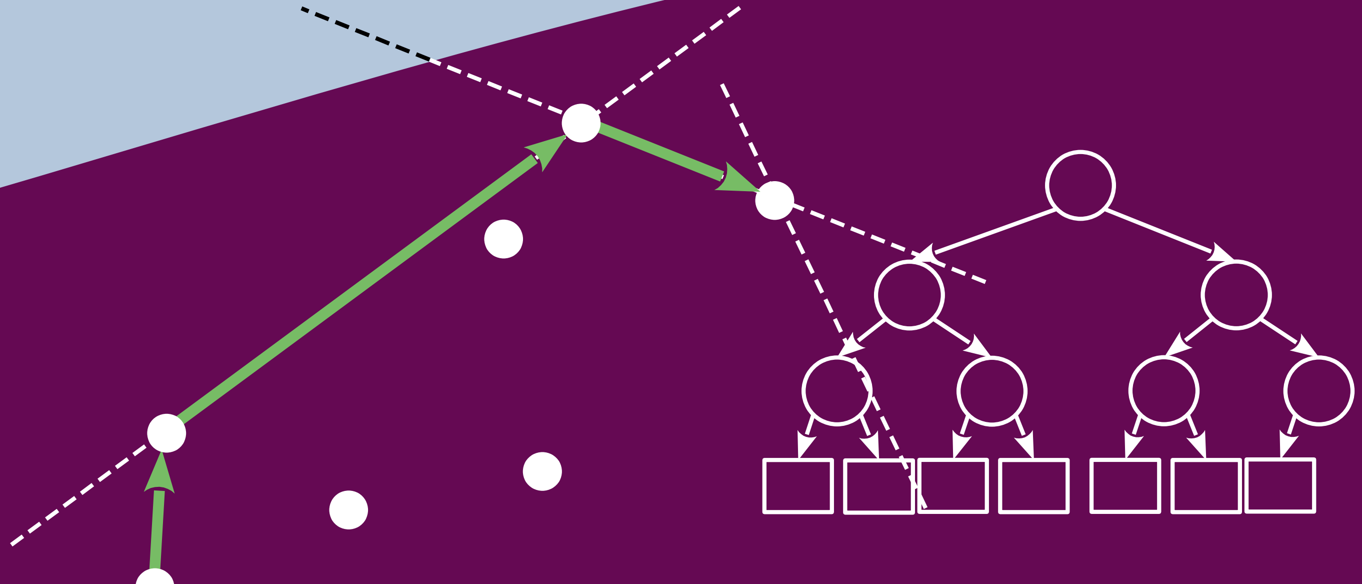
Advanced algorithms and structures data

12th lecture: Approximate algorithms







Approximate algorithms

- The basics
- MTSP – 2-approximation algorithm
- Vertex Cover – 2-approximation algorithm
- 0-1 knapsack – FPTAS

Lecture based on:

Script "Advanced algorithms and data structures", 2022.

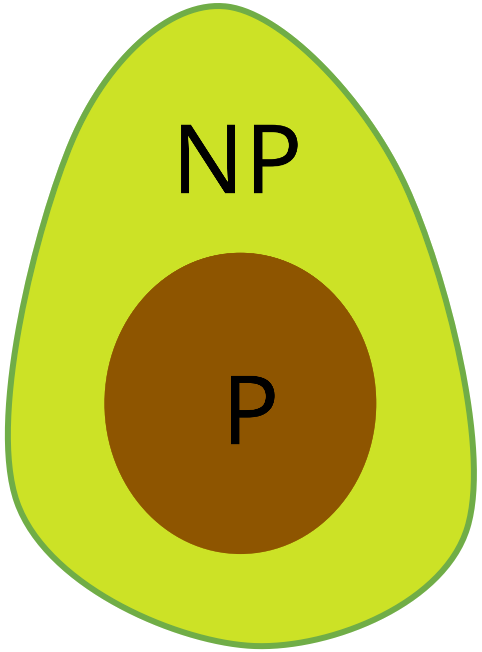
[WS11] DP Williamson, D. Shmoys, "The Design of Approximation Algorithms", 2011; **subchapters 1.1.-1.3, 2.4, 3.1**

Approximate algorithms?

- =?

- NP-hard discrete problems
    - We want to get the best possible solution in polynomial time
    - Guarantees loss of performance
    - Resource-quality tradeoff

- APX class (approximate)



Approximate algorithms?

- Common tools for the design of approximate algorithms
  - Greedy algorithms
  - Local search
  - Dynamic programming
  - Randomization
  - Quantization (rounding)
    - Basic
    - Adaptive
    - By accident
  - Convex optimization (eg LP)

α–approximate algorithm

- α–approximate optimization algorithm
  - Time polynomial
  - Solution zu**the worst**case within**factor α**from the optimum x*
    - To maximizez ≥ · *, < 1
    - To minimizez ≤ · *, > 1

PTAS

- Time polynomial approximation scheme
  - English polynomial-time approximation scheme (PTAS)
  - Family of algorithms $\{ \}$, for every > 0 there is a such that:
    - To maximize $(1 -)$-approximate algorithm 1
    - To minimize $(+)$-approximate algorithm

- A recipe, a meta-algorithm for the construction of approximate algorithms
  - Parameter

PTAS

- A recipe, a meta-algorithm for the construction of approximate algorithms
  - Parameter

- Polynomial with respect to the input problem,**not necessarily** with 1/

FPTAS

- Even more restrictive!

- Time fully polynomial approximation scheme
  - English fully polynomial-time approximation scheme (FPTAS)
  - PTAS such that the execution time of each runtime bounded from above by the polynomial u1/

Approximate algorithms

- Three key questions for every candidate

1. Correctness - a feasible solution?
2. Efficiency – polynomial time?
3. Quality - strict guarantees on the distance from the optimum?

Examples of INAPPROPRIATE problems

- Non-approximable in polynomial time (unless P=NP)
    - The general problem of the traveling salesman
    - Maximum click
    - Maximum independent set

Examples of suitable problems

- The traveling salesman's metric problem
  - minimization
- The vertex cover problem
  - minimization
- 0-1 knapsack problem
  - maximization

Metric Salesman Problem (MTSP)

- TSP + triangle inequality in distances
- NP-hard problem

- 2-approximation (2-MST heuristic)
  - Naive Eulerization of MST
- 3/2-approximation (Christofides, 1976)
  - Eulerization of MST à la CPP
- (3/2 −)approximation (Karlin et al., 2020)
  - Christofides, **random tree** instead of MST

Metric Salesman Problem (MTSP)

- NP-hard problem

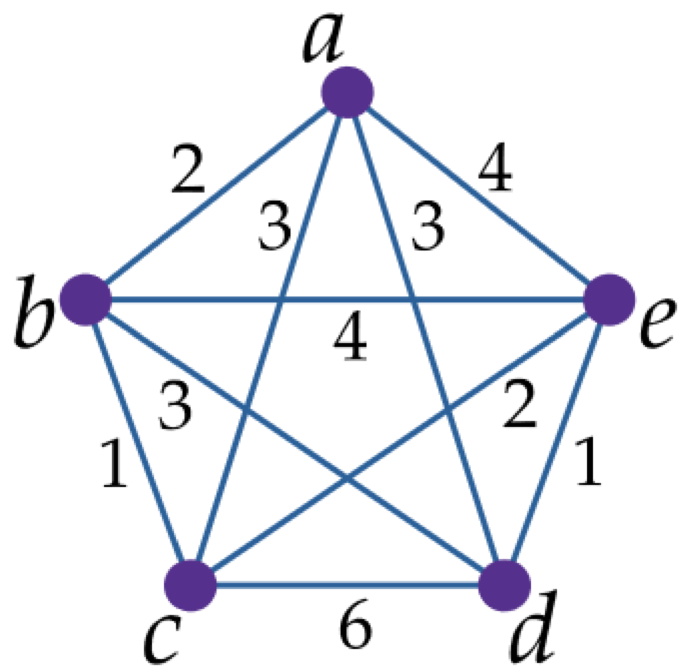
- The limit of approximability?
  - NP-hard to approximate with a factor $< 123/122$ (Karpinski et al. in 2013)

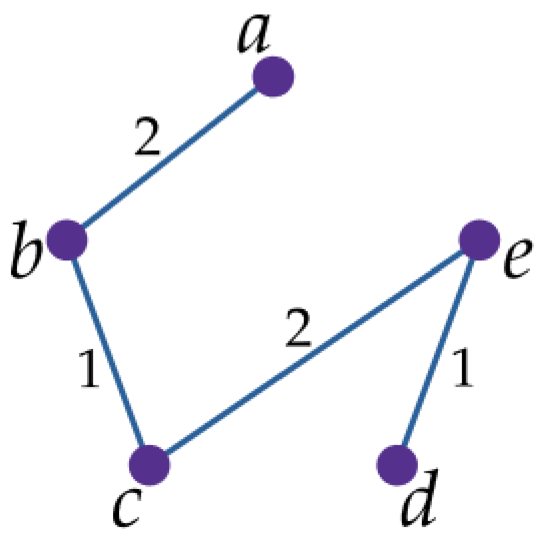
2-MST heuristics

Entrance: (, ×)

1. Find the MST in G, the weight ***x***
   O(|V|2)
2. DFS tour of all edges twice, write vertices in sheet L, length of tour ***2x*** O(|V|)
3. Filter L keeping only the first occurrences of vertices (short connection) and store in FL, the length of the tour ***z*** O(|V|)

2-MST heuristics - an example





1.MST

2. From the top a

L=[a,b,c,e,d,e,c,b,a]

3.*FL = [a,b,c,e,d]*

2-MST heuristics: analysis

- **Correctness**–FL contains each vertex once, and we interpret the ends as connected. A valid tour✓
- **Efficiency**–2-MST runs in polynomial time. ✓
- **Quality?**

2-MST heuristics: quality

- **Lemma 12.3.** For the weight x of the MST, x≤z holds.

- **Lemma 12.4.** 2-MST is a 2-approximation algorithm.✓

$$\leq \quad \leq 2$$

MST

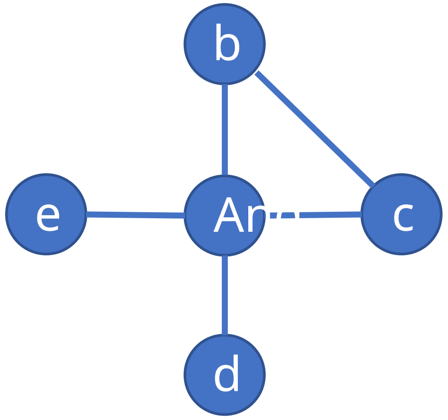
Inequality triangle

The peak cover problem

- Given an undirected graph (,)and peak costs : $\rightarrow \mathbb{R}$
- **Top cover**is ′ ⊆ such that for every edge in E one of the vertices in '
- The top cover of the minimum cost – 'such that the sum of costs in it is minimal

The peak cover problem

- The top cover of the minimum cost – 'such that the sum of costs in it is minimal



- A simple 2-approximation algorithm based on:
    - linear programming and
    - deterministic rounding of decimal numbers

The peak cover problem

- Modeled as an integer linear program

$$\min$$
$$\geq 1$$
$$\in 0.1\{\quad\}^{|\quad|}$$

-incidence matrix

$$= \begin{cases} 0 & \notin \,' \\ 1 & \in \,' \end{cases}$$

- NP-hard, optimal $*$

The peak cover problem - relaxation

- Continuous range

$$\min$$
$$\geq 1$$
$$\in [0,1] \quad | \quad |$$

Efficient solution!

- Optimum *, potentially decimal

$$^{*} \leq \quad ^{*}$$

Approximate algorithm -DetRoundLP

Entrance: $= (,), : \to \mathbb{R}$

1. $^{*}=$ LP relaxation solution

2. $= \quad (^{*})$

3. back $' = \{ \in \mid = 1\}$

DetRoundLP: analysis

- **Correctness**–the sum of two variables on the edges at least
  1. There must be at least one≥ 0.5
     - Rounding selects at least one incident vertex for each edge. Rolled top cover✓

- **Efficiency**–each step can be done in polynomial time (and solving LP)✓

DetRoundLP: quality

- **Lemma 12.2.** DetRoundLP is a 2-approximation algorithm.

$$ {}^{*} \leq \quad {}^{*} \quad \leq \quad \leq 2 \quad {}^{*} \leq 2 \quad {}^{*} $$

relaxation

optimality

rounding

relaxation

DetRoundLP: quality

- **Lemma 12.2.** DetRoundLP is a 2-approximation algorithm.

- A strict upper limit?
  - There are instances of graphs for which DetRoundLP produces a solution twice worse than the optimum

Vertex cover – limits of approximability

- [WS11] Ak o there is an α-approximate algorithm with < − ≈ . ,then P=NP

- If the unique games conjecture is true, the upper bound becomes

<

0-1 knapsack

- **NP-hard problem**

- Things $= \{1, \ldots ,\ \}$,each size $\in \mathbb{N}$,and values $\in \mathbb{N}$

- Backpack capacity $\in \mathbb{N}$
- Find the subset of things that fit in the knapsack and have the maximum sum of values

0-1 Knapsack

- **NP-hard problem**
- Dealing with DP –**pseudo-polynomial algorithm**
  - ( ) - numeric parameter
- Unary encoding – encoding with consecutive units
- Def. It's an algorithm**pseudo-polynomial**if performed in time to a polynomial input when the numeric part is encoded**unary**(rather than binary).

0-1 Knapsack

- If the numerical parameters are polynomial in n
  - Polynomial algorithm!!

- FPTAS – aggregation of numerical parameters into compartments
  - The number of compartments depends polynomially on n

0-1 Knapsack – new DP table!

- Table - columns of things, rows of knapsack values
- Cell [,] ->the least necessary cost that realizes value using some of the first things.

- Change required for approximate algorithm
  - Check what happens when you try the same trick with the following slides over a regular (costs, things) table

- The value of the most valuable thing = max

- -max value that can be achieved by a knapsack with n things and N

0-1 Knapsack – DP algorithm!

DPKnapsackByValue

1. [: , 1] = + 1
2. 0,1[= 0] 1, 1 =[min[1, 1 , 1) [ ]
3. h = 2, . . . ,
   1. [: ,] = [: , − 1]
   2. h = , ... ,
      1. [,] = min([, − 1], [− , − 1] +)
4. max{ : , ≤ } [ ]

2 ( )

pseudo-

polynomial on

- works by N

0-1 Knapsack – approximation

- We must limit N by a polynomial in n
- Quantization of values to units

BucketizedDP for knapsack – parameter

1. =

2. ′ = , ∀ ∈ [ ]

3. Fix modified problem using DPKnapsackByValue

Example 0-1 knapsack FPTAS

- Solve the 8 capacity backpack problem with the following 5 things**0.25-approximate algorithm**

	1	2	3	4	5
c	2	4	8	16	20
with	1	4	2	5	7

Example 0-1 knapsack FPTAS

- 0.25-approximate algorithm

- $= 8, \quad = 20$
- $= 0.75$
- $= \; = 0.75 \cdot 2 \; 0 = \frac{3}{5}$
- $' = \quad \left\lfloor - \right\rfloor, \forall \; \in$ → v'=[0,1,2,5,6]

	1	2	3	4	5
c	2	4	8	16 2	0
with	1	4	2	5 7	

0-1 knapsack FPTAS: analysis

- **Correctness**–DP returns a solution that satisfies the capacity constraint✓
- **Efficiency**– is transformed into after scaling . Complexity of the interlinked DP is $(\frac{}{3})$ ✓
- **Quality?**

0-1 knapsack FPTAS: quality

- **Lemma 12.5.**The presented algorithm is FPTAS for 0-1 knapsack.✓

- That is the solution is of value at least (1 −)from the optimal value

- Proof in script, based on**to the method of construction of the size of the compartments**

Conclusion

- For some problems we can find approximate algorithms
  - ***MTSP – 2-approximate, 1.5-approximate***
    - ***NP-hard for*** $< \frac{}{}$
  - ***Peak cover – 2-approximate***
    - ***NP-hard for*** $< \;.\;$ ***, MAYBE even*** $<$
  - ***0-1 knapsack – FPTAS***
    - ***A family of algorithms for everyone*** $\in (\;,\;)$
- For some problems we can't handle any
  - TSP, maximal clique and maximal anticlique

Conclusion

- We used the following techniques in the design of the algorithms
  - Quantization (rounding)
    - Adaptive input rounding - knapsack
    - Basic output rounding – top cover
  - Linear programming – vertex cover
  - Dynamic programming - knapsack
  - Greedy algorithms – MTSP
  - Local search - MTSP