COMP 446 / 546 ALGORITHM DESIGN AND ANALYSIS

LECTURE 15 APPROXIMATION ALGORITHMS

ALPTEKİN KÜPÇÜ

Based on slides of Shafi Goldwasser, Michael Goodrich, and Roberto Tamassia

DEALING WITH NP-COMPLETE PROBLEMS

- **Conjecture: P $\neq$ NP**
  - A widely believed conjecture is that no NP-complete problem has a polynomial time algorithm.
- **Alternatives:**
  - Run an exponential-time algorithm which always outputs the correct solution. (Useful if input is small.)
  - Run a polynomial-time algorithm which produces potentially incorrect solutions for some (or all) inputs.
    - Heuristic: a strategy for producing solutions with no guarantee for their correctness.
  - Run an approximation algorithm which always runs in polynomial time and produces a solution which is provably within a guaranteed approximation factor from the optimal solution.

OPTIMIZATION PROBLEMS

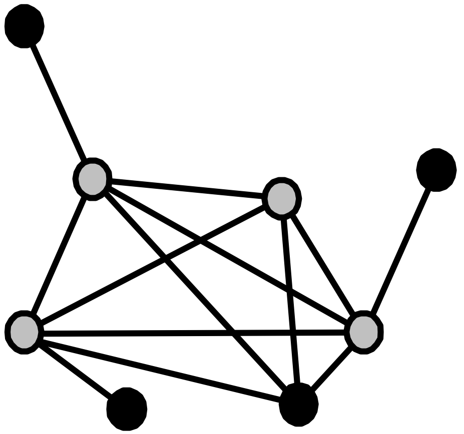
- **Optimization Problems**
  - We have some problem instance x that has many feasible solutions.
  - We are trying to minimize (or maximize) some cost function c(S) for a solution S to x.
- **Examples**
  - Find a minimum spanning tree of a graph
  - Find a smallest vertex cover of a graph
  - Find a shortest traveling salesperson tour in a graph

APPROXIMATION

- **An approximation produces a solution T**
  - T is a **k-approximation** to the optimal solution OPT if $c(T)/c(OPT) \leq k$
    - For maximization problems, use $c(OPT)/c(T)$
    - $k=1$ exactly when $c(OPT)=c(T)$
    - For example if $k=2$ and $c(OPT)=5$, then $5 \leq c(T) \leq 10$.
  - The approximation is provable.
- **A problem has a polynomial-time approximation scheme (PTAS) if it has a polynomial-time $(1+\varepsilon)$-approximation algorithm, for any constant $\varepsilon > 0$ (ε may affect the running time).**
  - Some NP-complete problems have PTAS for small constant k.
  - For some other NP-complete problems, computing any approximation with constant k is still NP-complete.

MIN VERTEX COVER

- **Input:** An undirected graph G=(V,E)
- **Problem:** Find a subset S of V that is a vertex cover of minimum size (every edge in E has at least one end point in S, and all other vertex covers contain at least |S| vertices)



- **Theorem:** MIN-VERTEX-COVER is NP-Hard
- **Proof:** Enough to show that the decision version is NP-Complete

MIN VERTEX COVER 2-APPROXIMATION

MIN-VERTEX-COVER-2 (G<V,E>)

```
C ← ∅
E' ← E
while E' ≠ ∅ do                          //some edges are not covered yet
    pick any edge e ∈ E' where e = (u,v)
    C← C ∪ {u,v}                         //add both u & v to C
    forall e ∈ incidentEdges(u) do
        E'.remove(e)
    forall e ∈ incidentEdges(v) do
        E'.remove(e)
return C
```

Polynomial-Time

MIN VERTEX COVER 2-APPROXIMATION

- **Theorem: The MIN-VERTEX-COVER-2 algorithm is a $k = 2$ approximation algorithm for minimum vertex cover problem. Namely, for every graph G, if OPT is the optimal (minimal) vertex cover and T is the vertex cover computed by the algorithm, then we have $|T| \leq 2\,|OPT|$**
- **Proof:**
  - First, the algorithm returns a vertex cover. ***Why?***
    - Since we iterate until every edge is covered.
  - A minimal vertex cover must include at least one vertex incident to each edge
  - The algorithm includes two.
    - At most twice the size.

MIN VERTEX COVER 2-APPROXIMATION (LP)

- **For each vertex:**
  - Create a variable x_i
  - Add constraint $0 \leq x_i \leq 1$ (think of $x_i = 1$ as picking the vertex)
- **For each edge (i.j)**
  - Add constraint $1 \leq x_i + x_j$
- **Objective function:** minimize $\sum_i xi$
- **The resulting linear program:**

minimize $\sum_i \boldsymbol{x_i}$

s.t. $\mathbf{0} \leq \boldsymbol{x_i} \leq \mathbf{1}$

$\mathbf{1} \leq \boldsymbol{x_i} + \boldsymbol{x_j}$

MIN VERTEX COVER 2-APPROXIMATION (LP)

- **MIN-VERTEX-COVER-LP (G)**
  - Solve the linear program:

    minimize $\sum_i x_i$

    s.t. $0 \leq x_i \leq 1$

    $1 \leq x_i + x_j$
  - LP solution (call OPT-frac) may have fractional x_i values. Add each vertex *i* to the vertex cover iff $\frac{1}{2} \leq xi$.
- **Theorem: The MIN-VERTEX-COVER-LP algorithm is a k = 2 approximation algorithm for minimum vertex cover problem.**
  - Observe that |OPT-frac| ≤ |OPT| ***WHY??***
  - The algorithm's solution T is a vertex cover. ***WHY??***
  - We have |T| ≤ 2 |OPT-frac|. Thus |T| ≤ 2 |OPT|. ***WHY??***

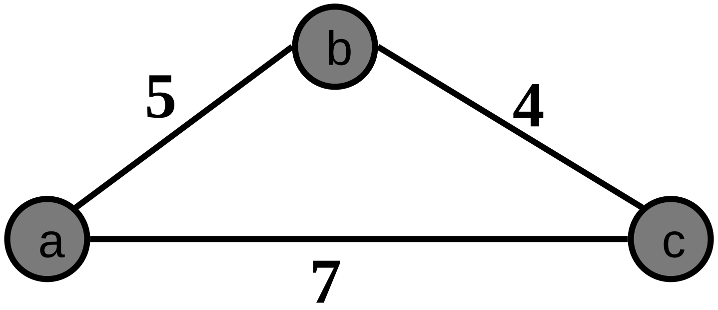
LP-BASED APPROXIMATION

- **General LP-based Method: Formulate a linear program that includes all solutions to an NP-hard problem, including fractional solutions (remember LP-relaxation). Then round up/down.**

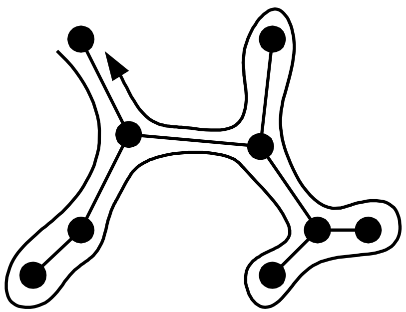
- **Interestingly, for Vertex Cover:**
  - No algorithm is known that achieves better than 2 - O(1 / sqrt(log n))
    - Not even 1.9
  - Theorem (Hastad): It is NP-hard to achieve an approximation algorithm with ratio better than 7/6.
    - Even an approximation is NP-hard.

SPECIAL CASE OF TSP

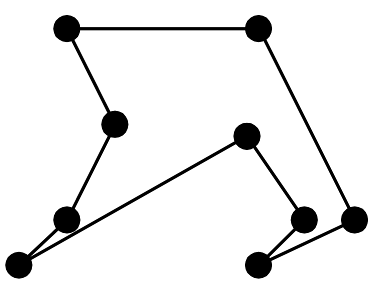
- **OPT-TSP: Given a complete, weighted graph, find a simple cycle of minimum cost that visits each vertex.**
  - OPT-TSP is NP-hard
- **Special case: Edge weights satisfy the triangle inequality**
  - w(a,b) + w(b,c) ≥ w(a,c)
  - Common in many applications



TSP SPECIAL CASE 2-APPROXIMATION



Euler tour P of MST M



Output tour T

TSPApprox(G)

Input: weighted complete graph G, satisfying the triangle inequality

Output: a TSP tour T for G

$M \leftarrow$ a minimum spanning tree for G

$P \leftarrow$ an Euler tour traversal of M, starting at some vertex s

$T \leftarrow$ empty list

for each vertex v in P //in traversal order

    if this is v's first appearance in P then

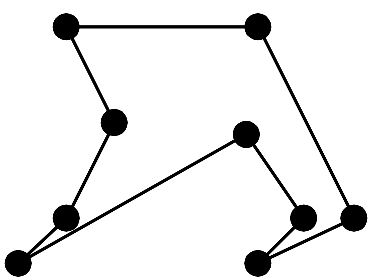
        $T.insertLast(v)$

$T.insertLast(s)$

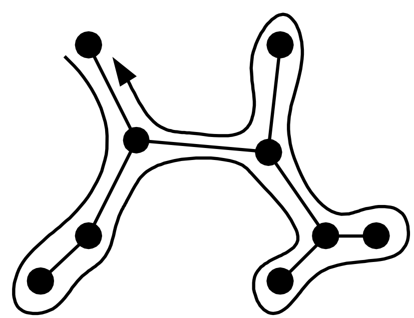
return T

TSP SPECIAL CASE 2-APPROXIMATION

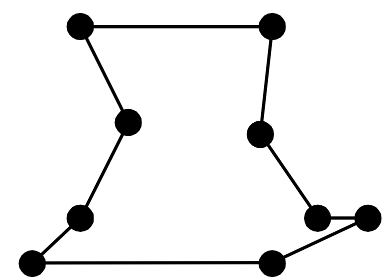
- **Proof of 2-approximation:**
  - Optimal tour is a spanning tour (must visit each vertex) → |M| ≤ |OPT|
  - The Euler tour P visits each edge of M twice → |P| = 2|M|
  - Each time we shortcut a vertex in the Euler Tour, we will not increase the total length → |T| ≤ |P|.
    - Remember the triangle inequality w(a,b) + w(b,c) ≥ w(a,c)
  - Therefore, |T| ≤ |P| = 2|M| ≤ 2|OPT|



Output tour *T*
(at most the cost of *P*)



Euler tour *P* of MST *M*
(twice the cost of *M*)



Optimal tour OPT
(at least the cost of MST *M*)

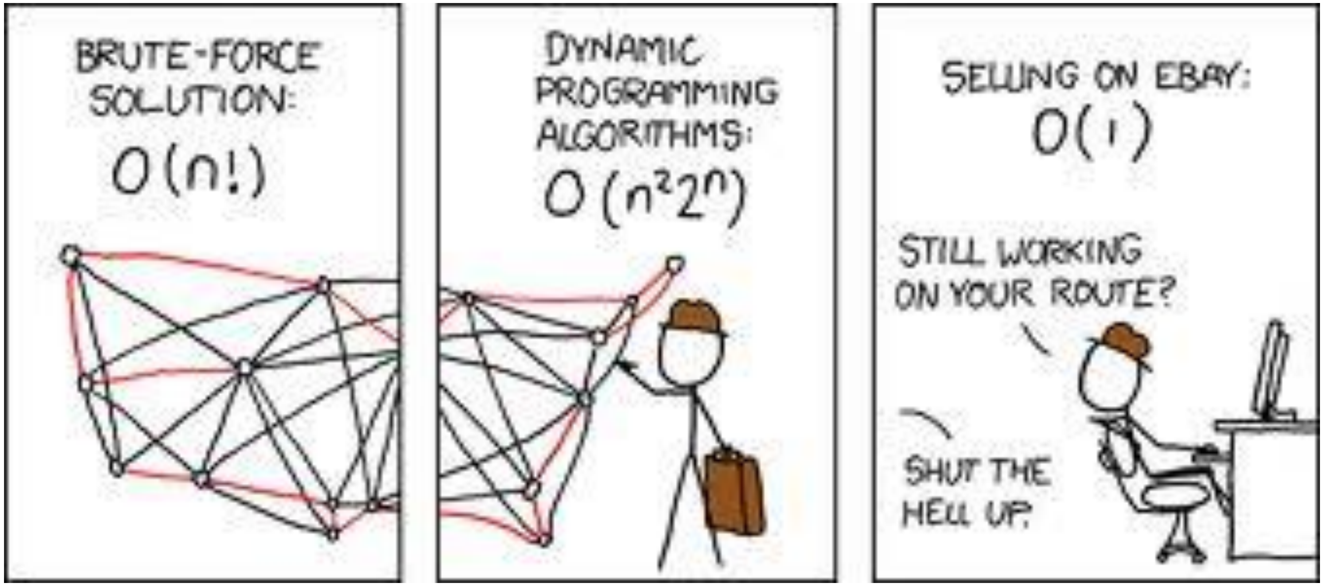
CONCLUSIONS

- **We can sometimes find efficient approximation algorithms for NP-hard problems.**
  - for CLIQUE $k = n / \log n$
  - for VERTEX-COVER $k = 2$
  - for SET-COVER $k = \log n$
  - for TSP special case $k = 2$
- **Conclusion: The fact that decision versions are computationally equivalent does not mean that problems can be approximated within the same factor!**

I WANT MORE !!!

- **To understand the underlying theory**
  - Take Computation and Complexity course
  - Join the reading/discussion group (email complexity@ku)
- **To understand how to make use of hard problems**
  - Take Modern Cryptography course
  - Join the reading/discussion group (email crypto@ku)
  - Check the website: https://crypto.ku.edu.tr
- **To work on similar topics**
  - Come talk with me
  - Join our group at any level (undergrad, master’s, phd, post-doc)
- **Please make use of the skills you gained in this course…**

TSP OPTIMAL SOLUTION



credit: xkcd