دانشگاه آزاد اسلامی واحد تبریز

نام درس: طراحی و تحلیل الگوریتم های پیشرفته

بخش: الگوریتم های تقریب

نام استاد: دکتر مسعود کارگر

MasoudKargar.ir

Approximation Algorithms

- Main goals of the lecture:
  - to understand the concepts of **approximation ratio, approximation algorithm,** and **approximation scheme**;
  - to understand the examples of approximation algorithms for the problems of **vertex-cover, traveling-salesman,** and **set-covering**.

MasoudKargar.ir

NP problems

- For some problems there are no known polynomial algorithms…
  - Hamiltonian-cycle problem:
    - Find a cycle in a graph that visits all vertices exactly once
  - Traveling-salesman problem (TSP):
    - Given a weighted undirected graph G=(V,E) and non-negative costs c(u,v), find a hamiltonian cycle with minimum cost.

SCIENCE

Santa Claus and the traveling salesman problem

Old Saint Nick has one night to deliver gifts to children around the world. To save time and wear and tear on his sleigh, Santa plots the shortest journey possible among thousands of cities. It sounds easy enough, and for Santa it probably is, but for scientists, who call this the traveling salesman problem (TSP), it is a challenge that grows increasingly difficult as more and more cities are added to the list.

Traveling salesman problem explained

[illegible]

30 Solution for Santa

[illegible]

Solving problem for more cities requires computers

[illegible]

Years ago

[illegible]

Everyday uses for TSP

[illegible]

13,509 A record

[illegible]

The hardware involved

[illegible]

MasoudKargar.ir

NP-complete problems

- There is a class of **NP-complete** problems
  - We look at the optimization variants of these problems
- All you need to know about NP-complete problems for this lecture:
  - no algorithms are known to solve them in polynomial time
    - $P \neq NP$ conjecture
  - They are related:
    - If we solved one in polynomial time, we could solve all of them in polynomial time, because we can convert input for any of them into input for any other in polynomial time (polynomial reduction)

MasoudKargar.ir

Coping with NP-completeness

- Do we surrender if we have an NP-complete problem?
  - Not so fast! Options:
    - Just use an exponential algorithm – either hope that the input is very small or that worst case manifests itself very rarely
      - use different **heuristics** to speed up search
    - Special cases maybe solvable in polynomial time
    - We may be able to find provably **near-optimal** solutions in polynomial time

MasoudKargar.ir

Simplified TSP

- Assumptions for simplified TSP:
  - The graph is complete
    - Each vertex has V-1 edges to all remaining vertices
  - **Triangle inequality** is satisfied:
    - For all u, v, w $\in$ V:
    - $c(u,w) \leq c(u,v) + c(v,w)$
- These are natural simplifications
  - For example: vertices – points in the plane, edge weights – euclidean distances between them

MasoudKargar.ir

Using Minimum Spanning Trees

- Can we start by computing something that is easy to compute and is somehow similar/related to the shortest tour?
  - **Minimum Spanning Tree**
  - How do we convert MST into a shortest tour?
  - If we perform a depth first search, we traverse all edges twice
  - Let’s just visit all vertices in the order of a preorder walk of the tree
    - Due to triangle inequality we are reducing the length

MasoudKargar.ir

Approximate MST

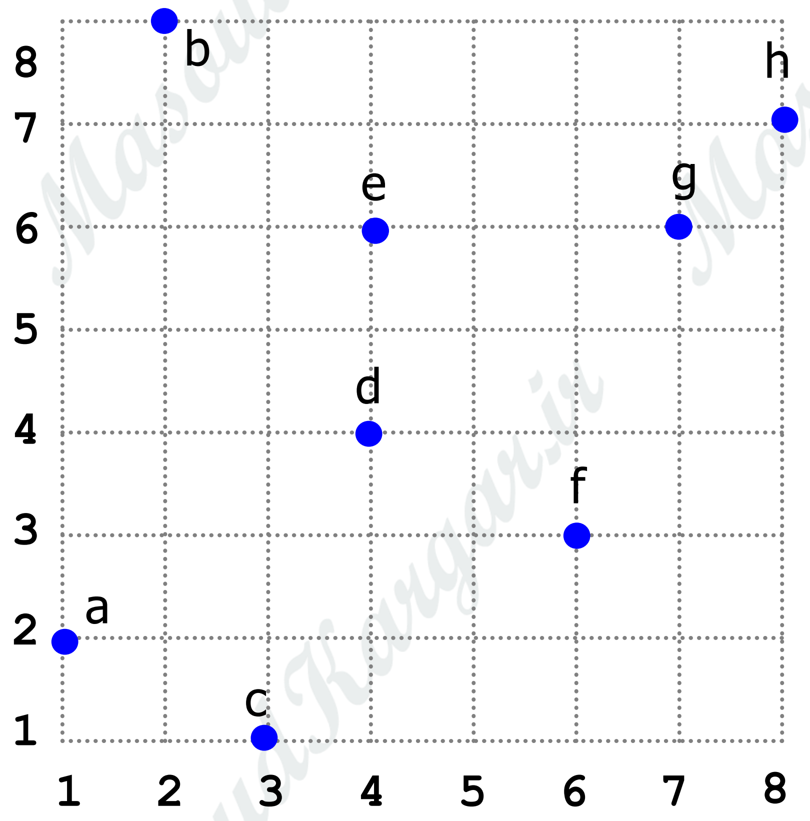
```
Approximate-TSP(G)
1 select a vertex r ∈ G.V to be a "root" vertex
2 T ← MST-Prim(G, r)   // compute an MST
3 let L be a list of vertices in a preorder tree walk of T
4 return the hamiltonian cycle H that visits the vertices in
  the order L
```

- What is its worst-case running time?

MasoudKargar.ir

Example

- Compute an approximate TSP tour:
    - Use a as a starting vertex



MasoudKargar.ir

Near-optimality

- How much worse is such a solution, when compared to an optimal one?
  - Observation: We can convert an optimal TSP H^* tour into a tree.
    - We get a lower bound – the cost of an MST T:
    - $c(T) \leq c(H^*)$
  - A full depth-first walk of the tree W visits all edges twice: $c(W) = 2c(T)$
  - This gives: $c(W) \leq 2c(H^*)$
  - The algorithm removes duplicate vertices by following direct edges between vertices to get H
    - This is possible because the graph is complete
    - This does not increase cost because of triangle inequality
  - Thus: $c(H) \leq c(W) \leq 2c(H^*)$

MasoudKargar.ir

Reflection

- We showed that our solutions are never more than twice worse than optimal
- How did we prove without knowing (constructing) an optimal solution?
  - We used a known structure (MST) to:
    - Use as a starting point in the algorithm
    - To prove the lower bound on the cost of an optimal solution

MasoudKargar.ir

Approximation algorithms

- Concepts, terminology:
  - **Approximation ratio $\rho(n)$**
    - For any input-size n, costs C and C^* of approximate and optimal solutions:
  - **ρ(n)-approximation algorithm**
  - **Approximation scheme**
    - Gets ε as input too, s.t. the scheme is a $(1+\varepsilon)$-approximation algorithm
    - Polynomial-time approximation scheme
  - **Fully polynomial-time approximation scheme**
    - Polynomial in both the input size and the ε

MasoudKargar.ir

No efficient ρ-approximation

- Do all NP-complete problems have polynomial ρ-approximation algorithms (where ρ is a constant)?
  - No! We can prove that the general TSP problem can not have a polynomial ρ-approximation algorithm, unless P = NP.
  - Proof by contradiction: if there is such an algorithm A we will use it to solve the hamiltonian-cycle problem:
    - We need to modify the input of a hamiltonian-cycle problem G = (V, E) to feed it to A.

MasoudKargar.ir

No efficient ρ-approximation

- Modifying the input of a hamiltonian-cycle problem:
  - Add edges to make the graph complete: G’=(V,E’)
  - Assign costs to edges:

$$c(u,v) = \begin{cases} 1 & \text{if } (u,v) \in E \\ \rho|V|+1 & \text{otherwise} \end{cases}$$

  - Two cases:
    - There is a hamiltonian-cycle in G
    - There is no such cycle
  - What each of these cases mean for the cost of the optimal TSP tour?
  - How this can be used to solve the hamiltonian-cycle problem with an algorithm A?

MasoudKargar.ir

Vertex-cover problem

- The problem:
  - Old network routers (vertices) must be changed to new ones which can “monitor” connections between routers (edges). To monitor a connection, one adjacent router is enough. What is the smallest amount of new routers needed to monitor all connections?
  - Formally: Given a graph $G=(V,E)$, find a minimum subset $V' \subseteq V$ such that if $(u,v) \in E$, then either $u \in V'$ or $v \in V'$ (or both).

MasoudKargar.ir

Maximal matching

- Again, find a concept that is easy to compute and is similar
  - We are covering edges
  - **Matching** is a subset of edges so that each vertex is adjacent to at most one edge
  - **Maximal matching** is a matching that is not a proper subset of any other matching
  - How do we compute it?
    - Let's just take edges one by one and try to include in the matching

MasoudKargar.ir

Approximate Vertex Cover

```
Approximate-Vertex-Cover(G)
1 C ← ∅
2 E′ ← G.E
3 while E′ ≠ ∅ do
4   Let (u,v) be an arbitrary edge of E′
5   C ← C ∪ {u,v}
6   Remove from E′ every edge adjacent to either u or v
7 return C
```

- Do the edges taken in step 4 constitute a maximal matching?
- Does it compute the vertex cover?
- What is its running time?

MasoudKargar.ir

Approximation ratio

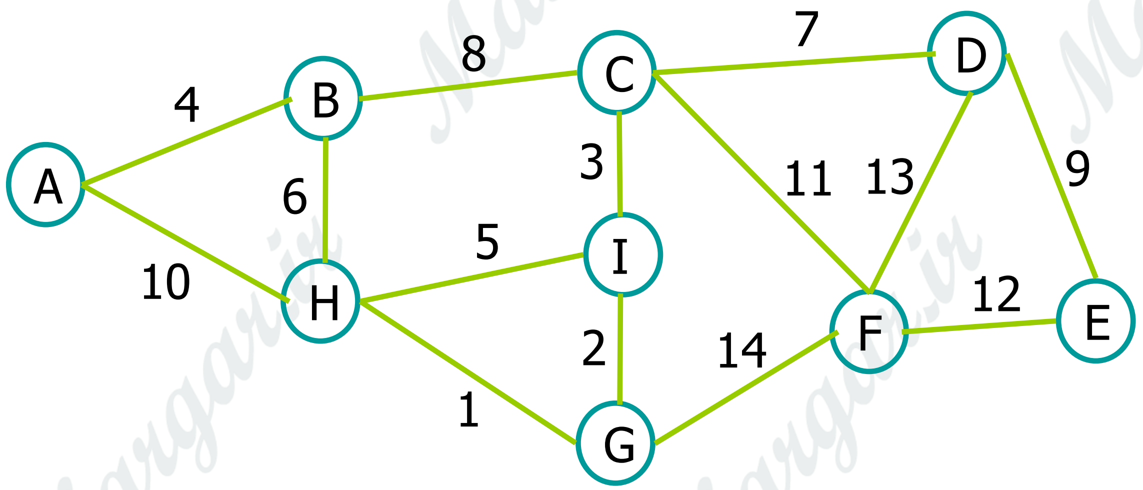
- What is the approximation ratio?
  - Observation: the size of the maximal matching A found in line 4 gives us a lower bound on the minimum vertex cover C^*: $|A| \leq |C^*|$ (Why?)
  - It is easy to see: $|C| = 2|A|$
  - Hence: $|C| = 2|A| \leq 2|C^*|$

- Conclusion: we have a 2-approximation algorithm

MasoudKargar.ir

Example

- Compute the approximate vertex cover



MasoudKargar.ir

Summary

- Good news:
  - You can find solutions that are “just” constant factor worse than optimal in polynomial time
  - You do that by:
    - Finding a similar but much easier problem
    - Just solving with a help of standard algorithm design techniques for optimization problems:
      - Dynamic programming, greedy algorithms, linear programming
- Bad news:
  - Not all problems can have such constant factor approximations in polynomial time.

MasoudKargar.ir