# Approximation Algorithms

Dr Deepak Garg

Computer Science and Engineering Department

Thapar University, Patiala

## Lecture contents

- We look for perfect answer always why approximation
- What is approximation ratio
- 2-approximation for vertex cover problem
- 2-approximation for TSP
- 1.5 approximation for TSP (conjectured optimal approximation)

## •Background

- The time and resources of researchers taken by P=NP question exceeds any other problem in computer science.
- Most of the people are coming out with proof P ≠ NP which is yet to be verified. So there are more voters of P ≠ NP theory.
- The scientists are not waiting for the final word on this question to come and then think about the next strategy.
- The Problems that are in question under this dilemma are so important in our daily life and for the society that these can not be left unsolved.

“I can’t find an efficient algorithm, but neither can all these famous people.”

## •Finding optimal solutions for specific categories

- Looking at the characteristics of the problem it should be explored whether it can be treated as some special case of a NP-complete problem or we can take advantage of any special feature or additional information given in the problem .
- Vertex cover is NP-complete but finding vertex cover for a bipartite graph can be done in polynomial time.
- Pseudo polynomial time algorithms may be better then their exponential counterparts.

  (An algorithm is said to run in pseudo polynomial time if its runtime is polynomial in the size of the input instance when the numbers in the input are represented in unary. Decimal 9 Binary 1001 Unary 111111111

## •Finding optimal solutions to NP-Complete problems

- Exponential algorithms may be adequate if problem size is small or problem is to be solved once.

- Some heuristics may be used which “work well in practice” but does not give any guarantees.

- Given the available resources, If exact solution is not available then we will try to find out a solution which is near to the optimal solution and this may be done in polynomial time.

- Difficult part is that at least we need to prove a solution’s value is close to optimum ( and to what extent), without even knowing what optimum value is!

## •Approximation factor

If $SV_A(i)$ : Solution value using Algorithm A on instance i
$SV_{OPT}(i)$: Solution value using Algorithm OPT on instance i
Assuming cost is positive always.
For Minimization problems : $SV_{OPT}(i) < SV_A(i)$
And for Maximization Problems : $SV_{OPT}(i) > SV_A(i)$

Approximation $\text{factor}_i$ or $\text{ratio}_i$ =
*$SV_A(i)/ SV_{OPT}(i)$ for Minimization*
*$SV_{OPT}(i)/ SV_A(i)$ for Maximization*
Approximation factor of A(n) = max{ approx. factors of all the instances of i of size n}

## •Objective of approximation

- Goal of the approximation algorithms is to design an algorithm such that approximation factor is as small as possible which means that we can go as close to the exact solution as possible in the polynomial time.
- It will be good if in some cases the complexity of the algorithm can be calculated in terms of the input size as well as the approximation factor.

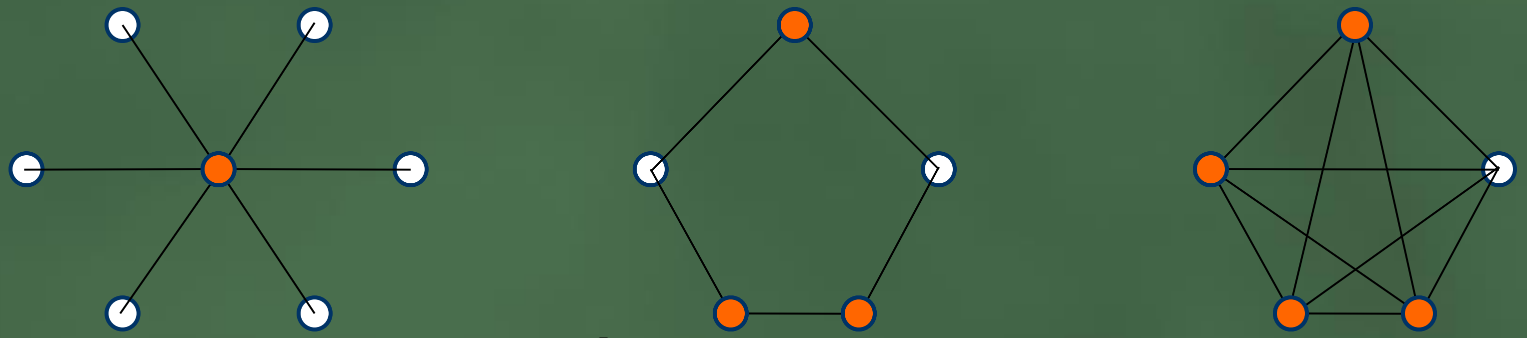
## •PTAS and FPTAS

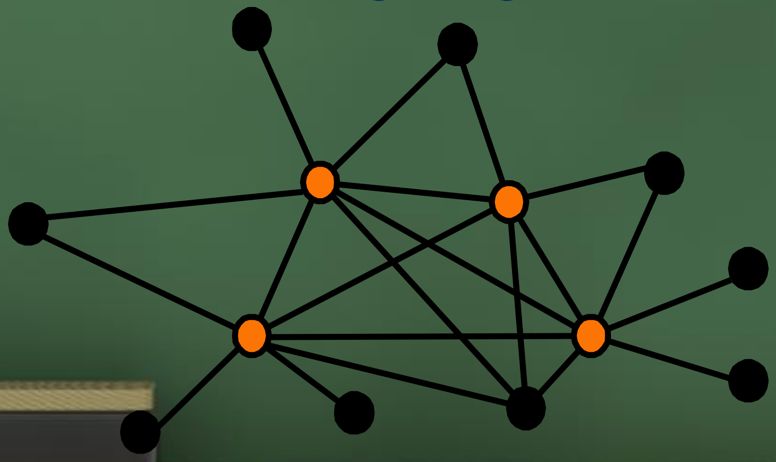
- Polynomial Time Approximation Scheme: It takes two arguments , an instance of the problem and a parameter $\varepsilon > 0$ and, in polynomial time, produces a solution that is within a factor $1 + \varepsilon$.
- For the TSP, a PTAS would produce a tour with length at most $(1 + \varepsilon)L$, where L is the length of the shortest tour.
- The running time of a PTAS is to be polynomial in n for every fixed ε but can be different for different ε.
- If the time is also polynomial in 1/ ε then it is called fully polynomial time approximation scheme (FPTAS).
- FPTAS for knapsack problem is $n^3/ \varepsilon$

## •Vertex cover

Input: graph G=(V,E) Output : A subset S of V, such that, for every (u,v) in E, u is in S or v is in S.

OPT-VERTEX-COVER: A subset with smallest size, which is NP-hard.





## •2-Approximation for vertex cover

1. Pick any edge {u,v} from set E of G
2. add both u & v to S
3. Delete u & v and all the incident edges from G
4. Repeat until any edge remains in G

Time taken by the above greedy algorithm is (V+E)

Output depends on the order we visit edges

## •Proof for 2-approximation vertex cover

- Every chosen edge e has both ends in S
- But e must be covered by an optimal cover; hence, one end of e must be in OPT
- Thus, there is at most twice as many vertices in C as in OPT.
- That is, S is a 2-approximation of OPT

- Best approximation ratio
- 2- (1/sqrt(logn)) [G Karakostas 2009 ACM trans on Algorithms]
- Best inapproximability
- (1+1/6) approximation algorithm unless P=NP [hastad  Elseveir Science Direct Theoretical Computer Science 97]

Further approximation itself will be NP-Hard

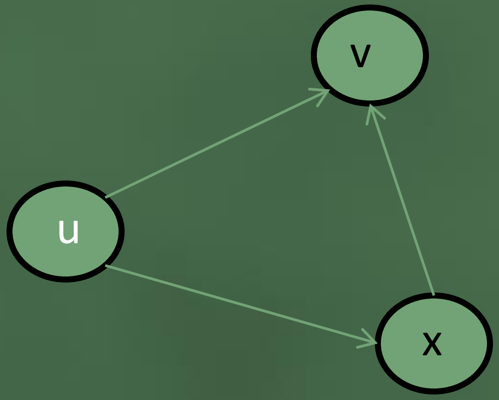
## •Travelling Salesman Problem (TSP)

- Input: Complete Graph of m vertices G= V,E and edge weight values w
- Output : cycle $V_1, V_2, V_3, \ldots\ldots V_m$ , $V_1$ visiting each vertex exactly once
- Objective : minimize weight of the cycle
- $w(V_1,V_2) + w(V_2,V_3) + w(V_3,V_4) + \ldots + w(V_{m-1},V_m) + w(V_m,V_1)$

- We consider the Metric TSP

## •Metric TSP

W(u,v) ≤ w(u,x) + w(x,v)  for all u,v, x
W(u,v) = Euclidean distance between points u & v
Or w(u,v) = shortest path weight from u-> v in graph G



## •2-Approximation Algorithm

1. Computer Minimum Spanning Tree T
2. Pick arbitrary Root vertex  for T
3. Output preorder traversal of  T

## •2-approximation for TSP

- In other words it is

. Construct the minimal spanning tree

- Walk around the tree visiting every tree edge exactly once
- Skip over repeats of vertices, taking direct edge to next vertex , guaranteed to be shortest by triangle inequality
- w of approximation algorithm ≤ 2 w(MST)
- ≤ 2 (OPT)
- Claim : w(OPT) ≥ w(MST)
- OPT = spanning cycle
- OPT minus one edge = spanning path of which is a spanning tree
- W(MST) ≤ w(spanning path) ≤ w(OPT)

- Best approximation ratio 1.5 [christofedes,CMU]
- Inapproximability 1+1/3812 [papadimitriou & yannakakis 93]
- General Approximation bound technique
- - Find lower bound on OPT
- - relate approximation algorithm to lower bound
- Other technique can be competing with the optimal

- Christofides Algorithm
  - There exists a 1.5-approximation algorithm for TSP
  - Find a minimum spanning tree T for Graph
  - Find a min cost perfect matching of odd degree nodes in T . Call it M
  - Do the union of spanning tree and matching edges. Call it G’
  - Find the Eulerian tour in G’
  - Find Short cut version of Eulerian Tour in E

- What is Matching
- matching or independent edge set in a graph is a set of edges without common vertices
- What is a Minimum-weighted Matching?

  It creates a MWM on a set of the nodes having an odd degree.
- Why odd degree?

  To fulfil the Property of Euler Cycle. Union of MST and MWM is Eulerian.

Imp Theorem: Every Graph has an even number of odd degree nodes

- Why 1.5 TSP?

  MST < Euler Cycle = MWM+MST <= 1.5 TSP

  (MWM = ½ MST)

## •Techniques for Approximation algorithms

- 1. Greedy Algorithms
- 2. Primal Dual Technique
- 3. Linear Programming and Rounding
- 4. Integer Programming

## •References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C, Stein, *Introduction to Algorithms, Third*
Edition, MIT Press, 2007.
[2] D. Hochbaum, editor, *Approximation Algorithms for NP-hard problems, PWS Publishing, Boston,*
1997.
[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NPCompleteness,*
W.H. Freeman & Co, 1979.
[4] V. Vazirani, *Approximation Algorithms, Springer, 2001.*