SOME APPROXIMATIONS

CS340

Why use an approximation?

- What properties does a good approximation algorithm have?

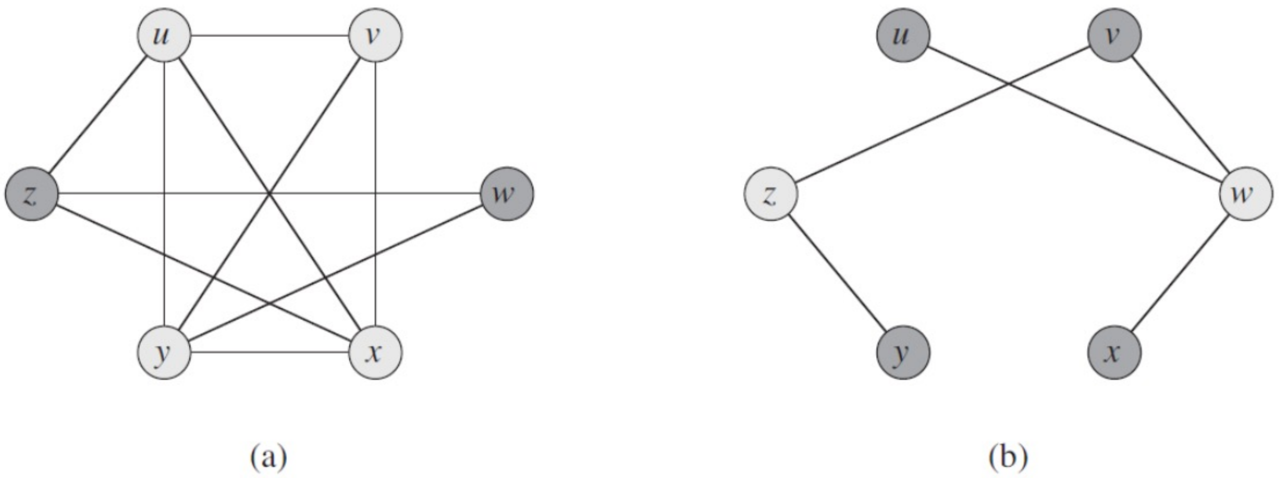
Subset Sum greedy approximation

- Bag will hold W weight, there are 3 items
- Sort by weight, largest to smallest…
  - W/2 + 1, W/2, W/2

- What is the greedy solution?
- What is the optimal solution?
- What is the worst thing that can happen?
- What can we say about this approximation?

Clique, Independent Set, Vertex Cover

- (a) shows a clique of size 4
- (b) is the complement of (a), and shows an independent set of size 4.
- (b)'s independent set of size 4 implies a vertex cover of size 2.



How about a greedy algorithm for vertex cover?

Approximations

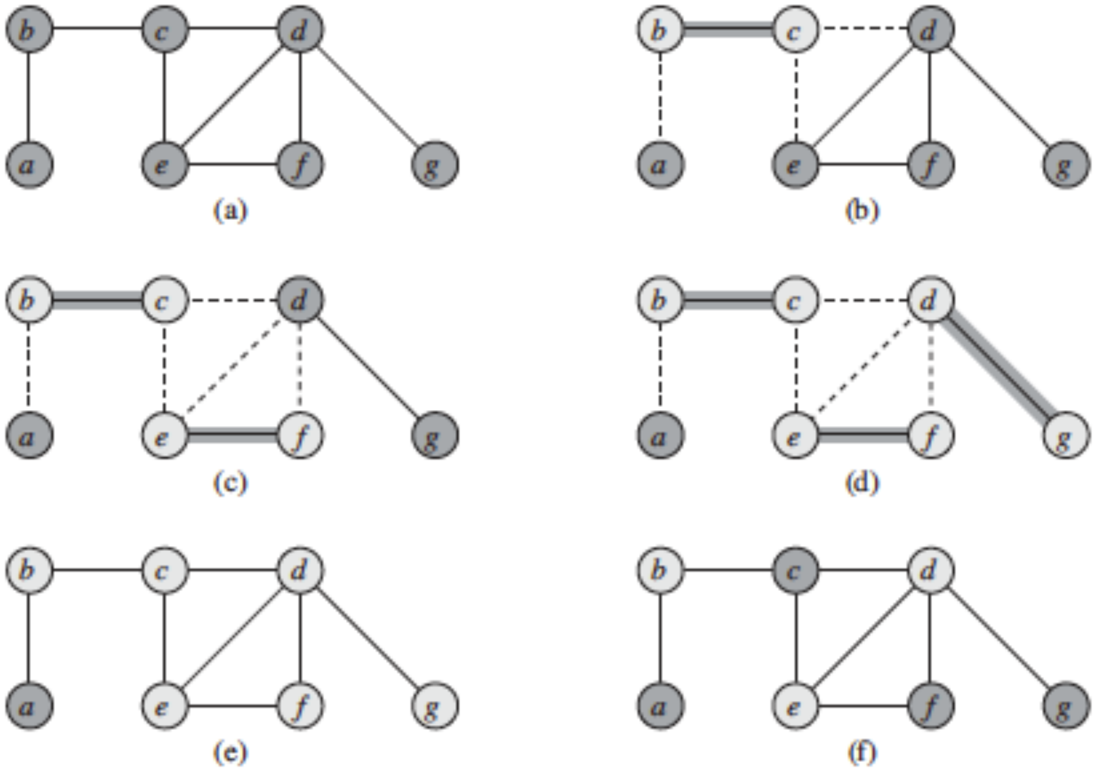
- Approximate Vertex Cover
  - Guaranteed to have at most twice the number of vertices as the true minimal vertex cover.

APPROX-VERTEX-COVER(G)

$C = \emptyset$
$E' = G.E$
while $E' \neq \emptyset$
    let (u, v) be an arbitrary edge of E'
    $C = C \cup \{u, v\}$
    remove from E' every edge incident on either u or v
return C

Approximations

- Approximate Vertex Cover
- (e) The 6-node vertex cover
- (f) The true 3-node vertex cover
- Notice the size of the vertex cover will always be 2x the number of edges picked.
- It is possible to do better.
- Is there an independent set?

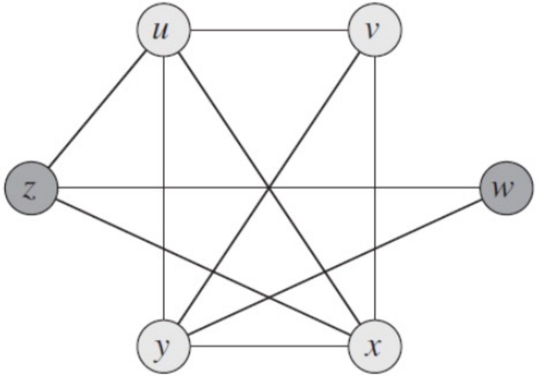


Approximate Vertex Cover

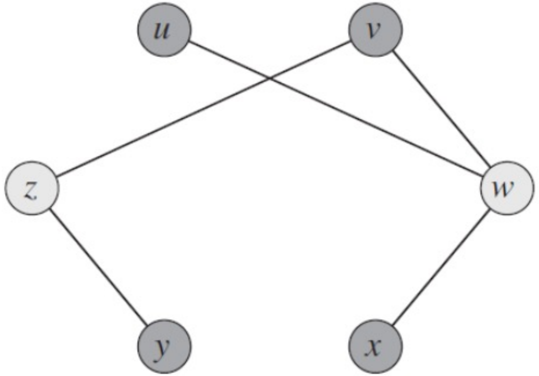
- Guaranteed to have at most twice the number of vertices as the true minimal vertex cover.
  - It is definitely a vertex cover. All edges are covered by a vertex.
  - A = set of edges picked by approximate vertex cover.
  - C* = optimal vertex cover; C = our approximate vertex cover
  - Any VC must include at least one of the endpoints of each edge in A.
  - No 2 edges in A share an endpoint, therefore:
  - |C*| >= |A|
  - Also, |C| = 2|A| as stated on previous slide!
  - Therefore, |C| <= 2|C*|

Can we approximate Clique?

- From your book: We know that the vertex-cover problem and the clique problem are complementary in the sense that an optimal vertex cover is the complement of a maximum-size clique in the complement graph. Does this relationship imply that there is a polynomial-time approximation algorithm for the clique problem?



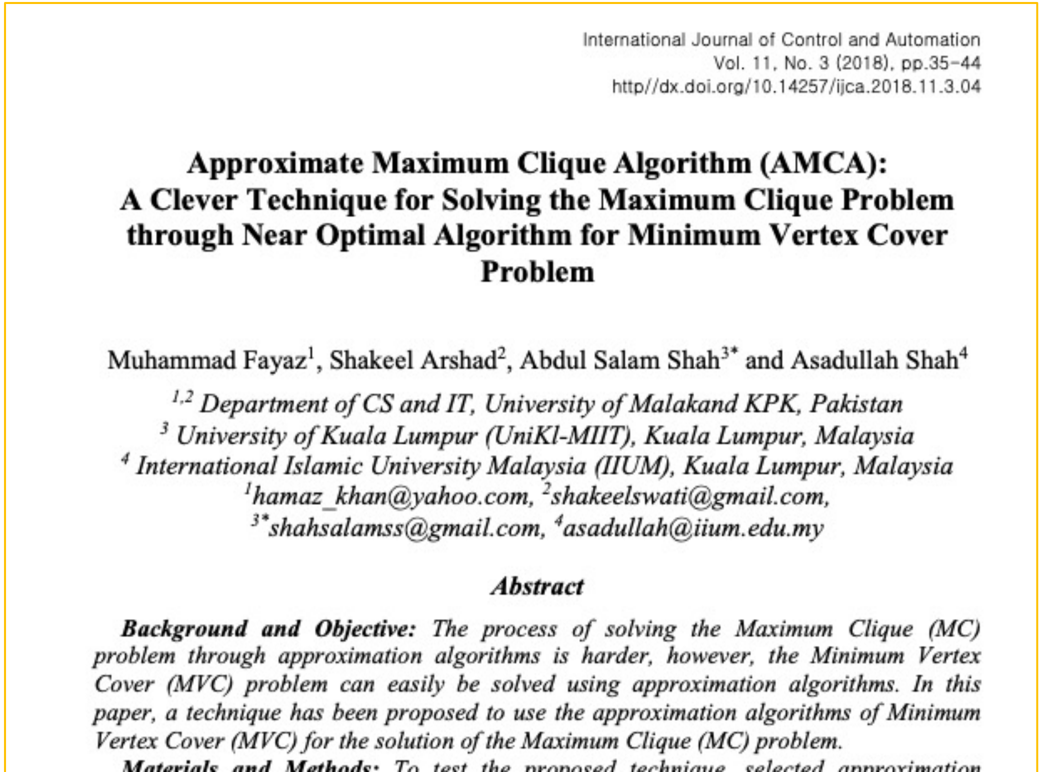
(a)



(b)

Google!

This paper gives no Mathematical bounds on the clique found



International Journal of Control and Automation
Vol. 11, No. 3 (2018), pp.35-44
http//dx.doi.org/10.14257/ijca.2018.11.3.04

Approximate Maximum Clique Algorithm (AMCA): A Clever Technique for Solving the Maximum Clique Problem through Near Optimal Algorithm for Minimum Vertex Cover Problem

Muhammad Fayaz[1], Shakeel Arshad[2], Abdul Salam Shah[3*] and Asadullah Shah[4]

[1,2] Department of CS and IT, University of Malakand KPK, Pakistan
[3] University of Kuala Lumpur (UniKl-MIIT), Kuala Lumpur, Malaysia
[4] International Islamic University Malaysia (IIUM), Kuala Lumpur, Malaysia
[1]hamaz_khan@yahoo.com, [2]shakeelswati@gmail.com,
[3]shahsalamss@gmail.com, [4]asadullah@iium.edu.my*

Abstract

Background and Objective: *The process of solving the Maximum Clique (MC) problem through approximation algorithms is harder, however, the Minimum Vertex Cover (MVC) problem can easily be solved using approximation algorithms. In this paper, a technique has been proposed to use the approximation algorithms of Minimum Vertex Cover (MVC) for the solution of the Maximum Clique (MC) problem.*

Materials and Methods: *To test the proposed technique, selected approximation*

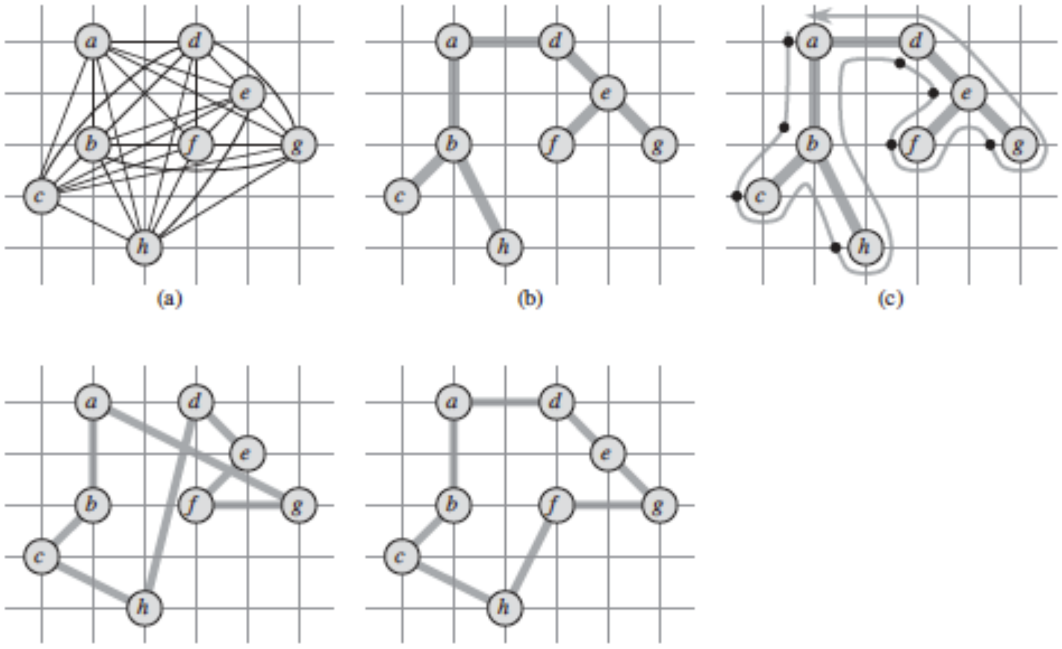
Approximate Traveling Salesman

- Guaranteed at most twice as long as the true shortest tour

```
APPROX-TSP-TOUR(G, c)
select a vertex r ∈ G.V to be a "root" vertex
compute a minimum spanning tree T for G from root r
    using MST-PRIM(G, c, r)
let H be a list of vertices, ordered according to when they are first visited
    in a preorder tree walk of T
return the hamiltonian cycle H
```

(Preorder is root, left, right)

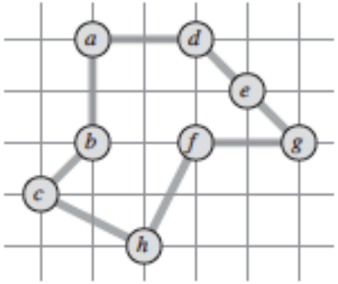
Traveling Salesman Approximation



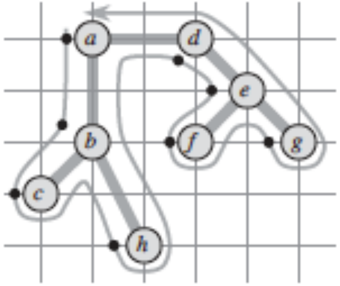
(a) (b) (c) (d) (e)

Traveling Salesman Approximation

- If we delete an edge from a tour (H*), it is a spanning tree.
- Cost of minimum spanning tree T is a lower bound: cost(T) <= cost(H*)
- Our inorder walk traverses each edge twice: cost(W) = 2*cost(T)
- Therefore cost(W) <= 2*cost(H*)



(e)



(c)

Approximate 3-SAT

- Guess?