Optimal Algorithm for Solving Vertex Cover Problem in Polynomial Time

Sharad Singh, Gaurav Singh, Neeraj Kushwah

Abstract:

Given an undirected, unweighted graph G = (V, E), the minimum vertex cover problem is a subset of whose cardinality is minimum subject to the premise that the selected vertices cover all edges in the graph. There are various other algorithms that follows heuristic based approaches, but using those approaches we are not able to find optimal solution always in the same manner Greedy algorithms solves the same problems but some problems have no efficient result, but up to a certain extend the Greedy algorithm may provide a solution that is near to optimal. In this paper we propose an Advance vertex cover algorithm to find optimal solution to the minimum vertex cover problem.

The proposed algorithm produce optimal solution with $O(n^2)$ time complexity, Optimality and time complexity is the main subject of this paper.

Index Terms— Vertex Cover Problem, Approximation Algorithms, Greedy Algorithm, NP Complete problem, Heuristic Approach, Proposed Algorithm, optimal algorithm.

Introduction

NP complete problem is if P class is in class NP and every problem p′ is reducible to P.In vertex cover problem we take an undirected graph (G=(V,E)) with V vertex and E edges.vertex cover problem will have a collection of set which contains the vertices and can cover all the edges of the graph..

Vertex cover

The vertex cover of a graph G is the subset of graph where every edge is covered. On the other hand we can say that vertex cover of a graph is the subset of graph that has at least one end point of every edge. By the use of vertex cover (subset of graph) we can traverse every edge of graph.

.**Definition:** "A vertex-cover of an undirected graph G= (V, E) is a subset of V`subset of $V$ such that if edge ($u$, $v$) is an edge of $G$ then either $u$ in $V$ or $v$ in $V$` (or both)."

Minimum Vertex Cover Problem

Minimum vertex cover problem is to find the smallest possible set of vertices that covers all the edges in the given graph G.

Definition: - "Given a graph G= (V,E), where V = { vertices in G }and E = {edges in G}.

The minimum vertex cover problem is to find a smallest subset of vertices V′(€ V), such that every edge has at least one endpoint in this subset ."

Literature Survey

In this paper we are trying to develop a new algorithm for vertex cover problem that can provide more optimal results in comparison of other algorithms. and our algorithm executing in polynomial time. Earlier we have seen many combinational optimizational problems like NP-Hard problems and there are minimal ways that we can develop an optimal and eficient algorithms for these problems. The minimum vertex cover comes under classical optimazation problems in computer engineering. this problem is example of non polynomial hard. so for this problem we have approximation algorithm.the vertex cover problem one of karps np hard problems and is therefore a np-complete problem in computational complexity theory.

[1]In practice, the minimum vertex cover problem can be used to model many real world situations in the areas of circuit design, telecommunications, and network flow and so on [1]. For example,[2] whenever one wants to monitor the operation of a large network by monitoring as few nodes as possible, the importance of the MVC problem comes into the rule[2].

Because of the computational intractability of the problem, many researches propose approximation algorithms for good solution of minimum vertex cover problem in a reasonable time. [2] An intuitive greedy approach for solving the problem is to successively select the vertex with the largest degree until all of the edges are covered by the vertices in V'. This straightforward heuristic is not a good one as demonstrated by [2].

Many researches present heuristic based solution for minimum vertex cover problem. Any approach that solves the problem without a

formal guarantee on the quality of the solution can be considered as a heuristics for the problem. Mostly Heuristic approaches not give optimal solution of the problem but some heuristics provide very good solution in practice.

Following are the papers that have heuristic approach to solve minimum vertex cover problem. Ira pramanick and jon G. Kuhl has proposed "A practical method for computing vertex covers for large graph". This paper is based on a general heuristic problem solving framework known as Parallel Dynamic Interaction or PDI[4]. In 1982m H. Papadimitriou and K. Steiglitz's research which is most popular now a days that us based on ant colony algorithm" "A pruning based ant colony algorithm for minimum vertex cover problem ". This paper fellow a meta-heuristic approach based on Ant Colony Optimization (ACO) approach to find approximation solution to the minimum vertex cover problem [1]. Another approximation based approach is given by Pietro S. Oliveto, Jun He, and Xin Yao in their research paper "Analysis of the (1+1)-EA for finding approximation solution to vertex cover problems". Evolutionary algorithms (EAs) are randomized search heuristics that have been widely used for solving combinatorial optimization problems since the 1970s [5].

Most of the researchers follow the greedy approach but the greedy approach not produces optimal solution. Greedy algorithms solve problems by making the choice that seems best at the particular moment. Many optimization (i.e. vertex cover problem) problems can be solved using a greedy algorithm. Some problems have no efficient solution, but a greedy algorithm may provide a solution that is close to optimal.

Existing greedy algorithm and heuristic approach

Existing approximation algorithm of vertex cover problem

1 C ← Ø

2 E′ ← E [G]

3 **while** E′ ≠ Ø

4 **do** let (u, v) be an arbitrary edge of E′

5 C ← C U {u, v}

6 remove every edge in E′ incident on u or v

7 **return** C

Above algorithm returns mostly approximate solution, that's why it is called approximation algorithm.

Greedy Algorithms of vertex-cover problem

1. C ← Ø

2. **while** E ≠ Ø

3. Pick any edge e €E and choose an end-point v of e

4. C ← C U {v}

5. E ← E \ {e € E : v € e}

6. **return** C

The above greedy algorithm returns good solution. [11]It is easy to find in some situations where this algorithm fails to yield a optimal solution. Greedy algorithm is not a 2-approximation [11].

Heuristic approach for vertex cover problem

1 C ← Ø

2 E′ ← E [G]

3 **while** E′ ≠ Ø

4 **do** let v be the highest degree vertex

5 C ← C U {v}

6 remove v and all edges incident to it

7 **return** C

The above heuristic approach algorithm return mostly optimal solution but not always.

But it always gives solutions better than greedy algorithm.

In this paper we propose an advance vertex cover algorithm that produces optimal solution in polynomial time. The time complexity of our algorithm is O (n^2).In this algorithm we prepare a list of vertices and traverse all vertices in the list. If all connected vertex from traversed vertex are exists in the list then delete traversed vertex but any connected vertex are not in list then no action performed on traversed vertex. Because of traversing and searching connected vertex from traversed vertex, the Time complexity of this algorithm is calculated as O(n^2).

Proposed algorithm

Step 1: Input all the vertices in the list V[N]

Where N is Total number of vertices.

Step 2: Sort the list V[N] according to degree of vertices.

Step 3: For each vertex u € V [N]

a) Delete the vertex *u* from the list V [N] if all the adjacent vertices from the *u are* exist in list V [N].
b) If vertex deleted then
N=N-1

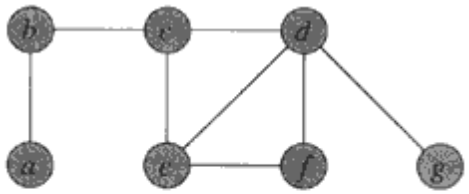
Step 4: Return V [N].

Explanation of proposed vertex cover algorithm

The first step of algorithm is used to input or store the data (vertices) of the graph. For taking input, algorithm uses an Array called V[n] list, where *n* is the total number of vertices. In the second step the V[n] list is sorted according to degree of vertices in ascending order by applying any sorting algorithm.

The third step is the main step of proposed algorithm where algorithm traverse each vertex in sorted list V[n] and delete the traversed vertex from the list V[n] if all adjacent vertex are exists in list V[n], else no need to delete the vertex. For implementing third step we can use any appropriate loop. Time complexity of this algorithm is based of this step because this step has maximum time complexity of all the steps. Time complexity of this step and this algorithm is O (n^2) because third step traverse all *n* vertices and search adjacent vertex of all traversed vertex. After third step we have a processed list V[n] which contain minimum vertex that covers all the edges in the given graph.

Example



(a)

Vertex	Adjacent Adages	Degree of vertex
a	b	1
b	a,c	2
c	b,e,d	3
d	c,e,f,g	4
e	c,d,f	3
f	e,d	2
g	d	1

Step 1: Input all the vertices in list V[n]

V[n]= {a,b,c,d,e,f,g}, n=7

Step 2: Sort the list V[n] according to degree of vertices

V[n] = {a,g,b,f,c,e,d}, n=7

Step 3: first iteration

Travers vertex *a* and Delete it from the list V[n] because only vertex *b* is adjacent from vertex *a* and vertex *b* is in the list V[n].So the list is

V[n]= {g,b,f,c,e,d }, n=6

Second iteration

Travers vertex *g* and Delete it from the list V[n] because only vertex *d* is adjacent from vertex *g* and vertex *d* is in the list V[n].So the list is

V[n]= { b,f,c,e,d }, n=5

Third iteration

Travers vertex *b*, there is no need to delete the vertex *b* because vertex b has two adjacent vertices, vertex *a* and vertex c, but in list V[n] the vertex *a* is not exist. So the list is

V[n]= { b,f,c,e,d }, n=5

Forth iteration

Travers vertex *f* and Delete it from the list V[N] because vertex f has two adjacent vertices, vertex *d* and vertex e and both vertex are in the list V[N] . So the list is

V[n] = { b,c,e,d }, n=4

Fifth iteration

Travers vertex *c* and Delete it from the list V[N] because vertex c has three adjacent vertices, vertex *b*,vertex d and and vertex e. All these vertices are in the list V[n] . So the list is

V[n] = { b,e,d } n=3

Sixth iteration

Travers vertex *e*, there is no need to delete the vertex e because vertex e has three adjacent vertices, vertex *c*, vertex d and vertex f , but in the list V[N] the vertex c and vertex f is not exist. So the list is

V[n] = { b,e,d }, n=3

Seventh iteration

Travers vertex *d*, there is no need to delete the vertex d because vertex d has four adjacent vertices, vertex *c*, vertex e, vertex f and vertex g , but in the list V[N] the vertex c , vertex g and vertex f is not exist. So the list is

V[n] = { b,e,d }, n=3

Step 4: Return the list V[n] because this is over optimum solution list.

V[n] = {b,e,d}, n=3.

RESULTS AND DISCUSSIONS

This section presents and discusses the analysis of all presented algorithms (in this paper) and complexity as shown below in the table. We illustrate the behavior of all studied

algorithms such as approximation algorithm, greedy algorithm, heuristic approach algorithm and proposed algorithm on the below graph.

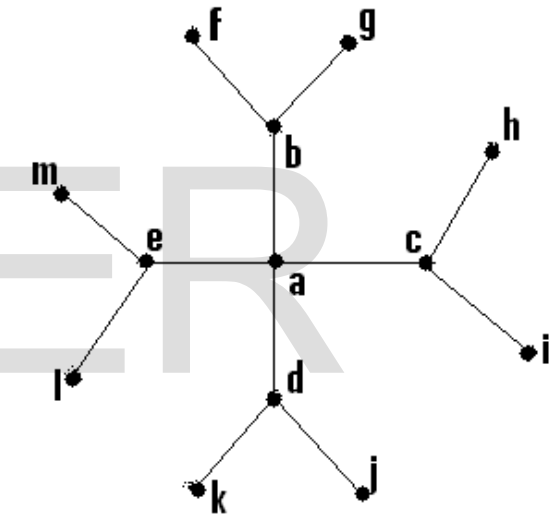


Figure 2: Graph (G) with 13 vertices

Vertex (V)	Number of edges connected to (V)	Connected vertices
a	4	b,c,d,e
b	3	a,f,g
c	3	a,h,i
d	3	a,j,k
e	3	a,l,m
f	1	b
g	1	b
h	1	c
i	1	c
j	1	d

k	1	d
l	1	e
m	1	e

Shows the comparison of all presented algorithms on graph (G)

Algorithm Type	Size of vertex cover	Solution set	Complexity	Optimality	Remarks
Approximation	8	{a,b,c,h,d,k,e,l}	O(V+E)	No	1. This gives different solutions but all solutions near to optimal. 2. This is a polynomial-time 2-approximation algorithm means that the solution returned by algorithm is at most twice the size of an optimal.
Greedy	5	{a,b,c,d,e}	O(V+E)	No	1. It is easy to find in some situations where this algorithm fails to yield a optimal solution. 2. Greedy algorithm is not a 2-approximation
Heuristic	5	{a,b,c,d,e}	O (logV)	No	1. Heuristic algorithm always gives solutions better than greedy.
Proposed Algorithm	4	{b,c,d,e}	$O(n^2)$, Where *n* is the no. of vertices	Yes	1. This algorithm always return optimal solution

Conclusion

One of the most challenging problems of the graph theory is the NP-Complete minimum vertex cover problem. In this paper, we introduced a simple but perfect algorithm called advance vertex cover problem that produced optimum solution of vertex cover problem with O (n^2) time complexity.

All the existing algorithms give approximation (near to optimum) solution of vertex cover problem, because most algorithm based on greedy approach and heuristic approach. But our algorithm produce optimal solution with O (n^2) time complexity

References

[1]Ali D Mehrabi, Saeed Mehrabi and Abbas Mehrabi, "A pruning based ant colony algorithm for minimum vertex cover problem".

[2] H. Papadimitriou and K. Steiglitz, (1982),"Combinatorial Optimization: Algorithms and Complexity", *Prentice Hall*.

[3] M. Karp, (1972), "Reducibility Among Combinatorial Problems", In R. E. Miller and J. W. Theater (eds), *Complexity of Computer Computations*, New York: Plenum Press, 1972.

[4] Ira pramanick and jon G. Kuhl, "A practical method for computing vertex covers for large graph".

[5] Pietro S. Oliveto, Student Member , IEEE, Jun He, Member IEEE and Xin Yao, Fellow, IEEE, " Analysis of (1+1)-EA for finding Approximate Solution to vertex cover problems.

[6] Faisal N. Abu-Khzamy, Rebecca L. Collinsz, Michael R. Fellowsx, Michael A. Langstonz, W. Henry Sutersz and Christopher T. Symonsz, "Kernelization Algorithms for the Vertex Cover Problem: Theory and Experiments"

[7] Toshimasa Watanabe, Satoshi Kajita and Kenji Onaga, "VERTEX COVERS AND CONNECTED VERTEX COVERS IN 3-CONNECTED GRAPHS" *Department of Circuits and Systems Faculty of Engineering, Hiroshima University,4-1, Kagamiyama I chome, Higashi-Hiroshima, 724 Japan*

[8] S. Balaji, V. Swaminathan and K. Kannan, "An Effective Algorithm for Minimum Weighted Vertex Cover problem" International Journal of Computational and Mathematical Sciences 4:1 2010

[9] Brenda S. Baker " Approximation Algorithm for NP- Complete Problems on Planner Graphs(preliminary version)", Bell Laboratories , Murray Hill, New Jersey 07974

[10] Thomas H. Cormen,Charles E. Leiserson,Ronald L. Rivest,Clifford Stein, "Introduction to Algorithms**",** Second Edition.

[11] David Avis - McGill University , Tomokazu Imamura -Kyoto University, " A List Heuristic for Vertex Cover"