International Journal of Control and Automation
Vol. 11, No. 2 (2018), pp.135-150
http//dx.doi.org/10.14257/ijca.2018.11.2.12

Approximate Methods for Minimum Vertex Cover Fail to Provide Optimal Results on Small Graph Instances: A Review

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 Kula 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

In this paper, a detailed literature review and comparative analysis, based on simplicity, efficiency and the run-time complexity of some well-known approximation algorithms for Minimum Vertex Cover (MVC) problem have been carried out. The key contribution of this paper is the provision of small benchmark graphs on which the given approximation algorithms fail to provide optimal results. The small benchmark graphs will help the researcher to evaluate efficient approximation algorithms. Generally, different terminologies and different styles have been adopted for writing pseudo code for different algorithms. To avoid such kind of difficulties, a uniform set of terminologies and pseudo code for each algorithm is provided in this paper, which will help researchers to easily understand the approximation algorithms for the Minimum Vertex Cover (MVC) problem.

Keywords*: Optimization Problem, Minimum Vertex Cover, Small Instance Graphs, Maximum Degree Algorithm, Vertex Support Algorithm, NP-Complete Problem*

1. Introduction

The graph plays an important role in the evaluation of algorithms in terms of their failure to provide optimal results. The graph is basically a collection of vertices connected by edges and same can be mathematically expressed as, G= (V, E), where vertices are denoted by 'V' and 'E' denotes edges. A graph can be used to model and simulate different real life problems, such as social systems and information systems. Graphs can also be utilized for the representation of computer networks and communication, computational devices, data organization, and the flow of computation [1-3].

A vertex cover of a graph is the set of vertices which cover all the edges of a graph and the minimum vertex problem is the smallest set of vertices in G= (V, E) to cover all edges in a graph. The minimum vertex cover problem has many applications in numerous areas, such as in Very-Large-Scale Integration (VLSI) design, Bioinformatics, Network Security, Scheduling, ATM machine Placing, Mobile Towers Placing, and Guards Placing [4-6]. These applications of minimum vertex cover have grabbed the attention of many researchers towards the problem, and consequently, numerous algorithms have been developed for tackling the MVC problem.

Received (October 12, 2017), Review Result (December 13, 2017), Accepted (December 23, 2017)
* Corresponding Author

ISSN: 2005-4297 IJCA

A problem which cannot be computed in polynomial time is said to be NP-problem, it takes exponential time to compute the large problem. The NP-complete problem has few most important properties, as first "it should be NP and should be reducible to the NP-complete problem in polynomial time" [7-8]. The minimum vertex cover is also NP-problem because it takes exponential time when the size of the problem is large. Richard Karp has proved in 1972, that the minimum vertex cover is NP-complete [9]. The NP-complete problems can be solved in two ways, *i.e.* through a complete or exact algorithm and by approximation algorithm. The complete algorithm always gives an exact solution of a problem, but the running time increases exponentially with the increase in the size of a problem, sometimes it may take billion or trillion years if computed utilizing currently available computation power [7, 10]. The minimum vertex cover is NP problem and the exact solution of an NP problem is applicable when the size of the problem is small and the solution is required with no time limitations. The exact solution for large instances is impossible in a reasonable amount of time [10]. The approximation algorithms always give the approximate solution of a problem in reasonable time, it is the best choice when the approximate solution is required in quick time for a large benchmark graph [11, 12]. In the last decade, the approximation algorithms gained the attention of the researchers to solve the NP-complete problem, and many attempts have been made for this purpose.

The approximation ratio is used to measure the performance of approximation algorithms, mathematically approximation ratio for MVC can be calculated by (1).

$$\rho i = \frac{Ai}{Opti} \geq 1 \tag{1}$$

Where Ai represents the solution returned by an approximation algorithm and the Opti is the optimal solution of a problem. The value of ρi must be always greater or equal to 1 ($\rho i \geq 1$), $\rho i = 1$ indicates that the solution returned by an approximation algorithm is optimal and the deviation from 1 indicates the poor solution of an approximation algorithm [13-14].

The problem of finding the best solution in all feasible solutions is known as optimization problem [11]. The graph can be covered by using different combinations of vertices, but in the minimum vertex cover problem, we cover all the edges by selecting the smallest set of vertices. The vertices other than MVC are MIS vertices in G, such as MIS: = V-MVC and similarly, MVC: = V-MIS. The maximum clique can be converted to MVC by taking the complement of a graph and subtract from G those vertices which are in the clique, so the remaining vertices will form an MVC set of G, such as, MVC:= V- w(G') [9], [14].

Further for the better understanding of the algorithms the terminologies are discussed in detail as:

1.1. Terminologies

Let G = (V, E), be an undirected graph, where elements of "V are called vertices, V = {$v_1, v_2, v_3 \ldots v_n$} and elements of E are called edges, E subset {{u, v}: u, v ∈ V}, and let |V| = n and |E |= m, p= |N (v)|", [15].

"Neighborhood of a vertex: For each v ∈ V, N (v) = {u ∈ V /u is adjacent to v} and (N[v] = v U N (v)", [16].

Degree of a vertex: The degree of a vertex is denoted by deg (v), where v ∈ V.

Maximum degree of a vertex: It is denoted by Δ (v) where v ∈ V, return the maximum degree of a vertex in G= (V, E).

Minimum degree of a vertex: The minimum degree vertex in G is represented by δ in G= (V, E).

Support of a vertex: The sum of the degree of vertices adjacent to a vertex v ∈V is known as the support of that vertex, *i.e*. s (v) = support (v) =$\sum_{u \in N}(v) d(u)$.

Maximum support of a vertex: The maximum support of a vertex u ∈ v having maximum support denoted by Ms (u) = maxsupport ($\sum_{u \in N}(v) d(u)$).

Minimum support of a vertex: The minimum support of a vertex u ∈ v having minimum support denoted by ms (u) = minsupport ($\sum_{u \in N}(v) d(u)$).

Tie: The vertices having the same degree or equal deserve for being a vertex in vertex cover.

Minimum sum of degrees: Minimum sum of degree GetMinVertex ( ).

Bold circle: Bold circle in a graph represents the vertex in vertex cover.

In this paper the survey of some well-known approximation algorithms is carried in detail. These algorithms are chosen based on simplicity, optimality, and low time complexity, with each algorithm we have also carried out a uniform pseudo code for better understanding for readers. A small graph is also given with each algorithm on which the given algorithms get fail, this will help researchers to design a better approximation algorithm for MVC.

The rest of the paper is organized as the section 2 contains literature review, the section 3 contains a comparative analysis of the algorithms and finally the conclusion of the study is presented in section 4.

2. Literature Review

Clarkson *et al*. in [17], proposed efficient approximation algorithm for minimum vertex cover, it is a simple and fast algorithm. The maximum degree algorithm (MDG) is based on greedy approach, it is the modified version of the set cover problem developed by Chavatal in 1979 [18]. It incorporates and adds those vertices to the MVC which have a maximum degree, among all vertices. The pseudo code of the maximum degree algorithm is as in Figure 1.

Maximum Degree Algorithm (MDG)
Input: G= (V, E)
Output: MVC
Begin:
1. C ←∅;
   //initially no vertex is in the vertex cover set.
2. While E ≠∅ do {
3. n ← |G(V,E|
4. for i ← 1 to n {
5. d (v_i)
   // calculate the degree of each vertex.
6. }
7. u← Δ (G)
   // return a vertex having the maximum degree and assigned to u.
8. V ← V − {u}.
   // deleted the maximum degree vertex u from the graph.
9. C←C ∪ {u}
   // add u to vertex cover.
10. Go to 2. }
11. Return MVC

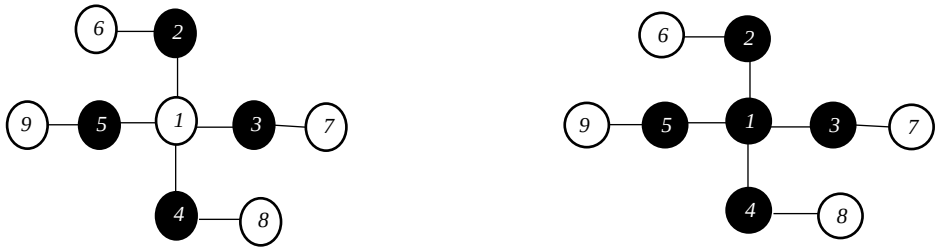
End

Figure 1. Pseudocode of Maximum Degree Algorithm

The worst runtime complexity of the MDG is O (E^2), which indicates the computation power of algorithm. The disadvantage of the MDG algorithm is that it fails even on small benchmark graphs as on G_1 the given graph fails to provide optimal result.

By applying MDG on G_1 the degree of each vertex is calculated, the maximum degree of the given graph is of vertex 1 which is 4. So according to the MDG algorithm, vertex 1 is included in MVC and it is removed from G_1. In the second step the G_1 becomes $G_1 = G_1 - v_1$, the degree of each vertex is again calculated, the maximum degree is 1 and all the vertices in G_1 have degree 1, so we selected vertex 2 randomly and included in VC and the G_1 become $G_1 = G_1 - v_2$. This process continues until no edge remains in G_1.

G1



Graph 1. (a). Optimal MVC ={2, 3, 4, 5} (b). MDG MVC = {1, 2, 3, 4, 5}

The whole process of MDG is illustrated in Table 1 where the V represents the vertices of G_1 which have the degree greater than zero and E represents the edges which have not yet been covered, d (V) represents the degree of each vertex of G1, u represents the vertex having a maximum degree. A number enclosed in {} represents the vertex or vertices, and a number(s) that is not enclosed in {} represents the degree of vertex or vertices.

Table 1. Working Mechanism of MDG Algorithm on G1

Sr. No.	V	E	d(V)	u	C
1	{1,2,3,4,5,6,7,8,9}	{(1,2), (1,3), (1,4), (1,5), (2,6), (3,7), (4,8), (5,9)}	4,2,2,2,2,1,1,1,1	4{1}	{1}
2	{2,3,4,5,7,8,9}	{(2,6), (3,7), (4,8), (5,9)}	1,1,1,1,1,1,1,1	1{2}	{1,2}
3	{3,4,5,7,8,9}	{ (3,7), (4,8), (5,9)}	1,1,1,1,1,1	1{3}	{1,2,3}
4	{4,5,8,9}	{ (4,8), (5,9)}	1,1,1,1	1{4}	{1,2,3,4}
5	{5,9}	{5,9}		U6V	{1,2,3,4,5}
6	{}	{ }	-----	------	{1,2,3,4,5}

Balaji *et al*. in [5], proposed Vertex Support Algorithm (VSA) which is an approximation algorithm. In this algorithm, a new data structure vertex support is introduced. The vertex support is the sum of degrees of all adjacent vertices of a vertex. The support of all vertices is calculated and a vertex having maximum support among all vertices is added to MVC. This algorithm is also based on greedy strategy, the computation complexity of this algorithm is O (En^2) [5]. The pseudocode of the VSA algorithm is given in Figure 2.


```
Vertex Support Algorithm (VSA)
Input: G= (V, E)
Output: MVC
Begin:
1.      C ←∅;
                    //initially the vertex cover is empty.
2.      while E ≠ϕ do {
3.      n ← |G|
4.      for i ← 1 to n {
5.      d(vi )
                    //calculate the degree of each vertex and assign to an array A.
6.      SV[i] ← s (vi)
                    // calculate the support of each vertex and assign to an array SV.
7.      } u ← Ms( SV[i] ))
                    // find out the maximum support vertex and assign to u.
8.      G← G \{u}
                    // delete the maximum support vertex u from the graph.
9.      C←C ∪ {u}
                    // add the maximum support vertex to the vertex cover.
10.     } return C.
End
```

Figure 2. Pseudo Code of Vertex Support Algorithm (VSA)

The run time complexity of VSA is O (EV^2). Like the MDG algorithm, it also fails on small benchmark instances and it fails on G_1

By applying VSA on G_1, first the degree and then the support of each vertex is calculated, the maximum degree of G_1 is of vertex 1, which is 8. So according to MDG algorithm vertex 1 is included in VC and it is removed from G_1, so G_1 becomes $G_1= G_1-v_1$. In the second step the, the degree and maximum support of each vertex are again calculated of new G_1. In this case the maximum support of G_1 is 1 and all the vertices in G_1 have support 1, so we selected vertex 2 randomly and included in VC and now the G_1 become $G_1=G_1-v_2$. This process continues till the graph turn out to be blank. The whole process of VSA is shown in the Table 2.

Table 2. Working Mechanism of the VSA by applying it on G1 in Graph 1

Sr. No.	V	E	d(V)	u	C
1	{1,2,3,4,5,6,7,8,9}	{(1,2), (1,3), (1,4), (1,5), (2,6), (3,7), (4,8), (5,9)}	4,2,2,2,2,1,1,1,1	4{1}	{1}
2	{2,3,4,5,7,8,9}	{(2,6), (3,7), (4,8), (5,9)}	1,1,1,1,1,1,1,1	1{2}	{1,2}
3	{3,4,5,7,8,9}	{ (3,7), (4,8), (5,9)}	1,1,1,1,1,1	1{3}	{1,2,3}
4	{4,5,8,9}	{ (4,8), (5,9)}	1,1,1,1	1{4}	{1,2,3,4}
5	{5,9}	{5,9}		U6V	{1,2,3,4,5}
6	{}	{ }	-----	------	{1,2,3,4,5}

Imran *et al* in [19] modified the vertex support algorithm which uses the same data structure introduced by VSA. MVSA calculates support of every vertex of a graph to find the minimum support vertices. All adjacent vertices to the minimum support vertex are then located and the vertex having minimum support is selected as the candidate vertex for MVC. The decision making process does not involve any extra computation complexity and its run time complexity is O (EV^2log v). Like

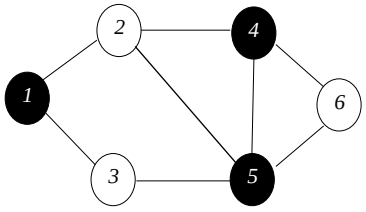
MDG and MVSA it also fails at small instances. The pseudo code of the MVSA is given in Figure 3.

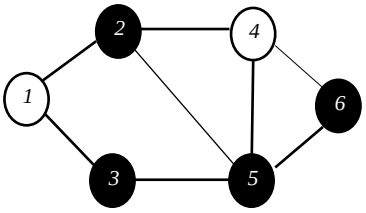
Modified Vertex Support Algorithm (MVSA)
Input: G= (V, E).
Output: MVC
Begin:
1. C ←∅;
//initially the vertex cover is empty.
2. while E ≠ϕ do {
3. n ← |G|
4. for i ← 1 to n {
5. d(v_i)
// calculate the degree of each vertex and assign to an array A.
6. SV[i] ← s (v_i)
// calculate the support of each vertex and assign to an array SV.
7. x ← min(SV[i])
// calculate the minimum support vertex in G and assign to x.
8. ad_min_supp(x)
// find out all the adjacent vertices of the minimum support vertex.
9. u ← min(ad_min_supp(x))
// find out the minimum support vertex in all adjacent vertices of MSV.
10. V←V- u
11. C ← u
12. Return C.
End

Figure 3. Pseudo code of Modified Vertex Support Algorithm (MVSA)

When we apply the MVSA on a given graph G_2, like VSA, the degree and support of each vertex are calculated, then the minimum support vertex is located, which is the vertex 1 and it has minimum support among all vertices of G_2, and the adjacent vertices of the minimum support vertex are found which are 2 and 3. The support of vertices 2 and 3 are 9 and 6 respectively, the vertex 3 has minimum support hence, it is included in vetex cover and removed from G_2, *i.e.* $G_2=G_2-v_3$. In the second step, the minimum support vertex is again 1 which has minimum support among all vertices in G_2-v_3. The adjacent vertex of vertex 1 is 2, so it is included in VC, now the $G_2 = G_2-v_3-v_2$.

G2





Graph 2. (a). Optimal MVC ={1,4,5} (b). MVSA MVC ={2,3,5,6}

The whole process of MVSA is presented in the Table 3. The S(V) represents the support of each vertex, ms(V) represents the minimum support vertex and ad_min(ms(G)) represents the vertices adjacent to the minimum support vertex.

Table 3. Deployment of MVSA on G2

Sr. No.	V	E	S(V)	ms(G)	Ad_min (ms(G))	u	C
1	{1,2,3,4,5,6}	{(1,2), (1,3), (2,4), (2,5),(3,5), (4,6), (5,6)}	{5,9,6,9,10,7}	5{1}	{2,3}	3	{3}
2	{1,2,4,5,6}	{(1,2), (2,4), (2,5), (4,5),(4,6),(5,6)}	{3,7,8,8,6}	3	{2}	2	{2,3}
3	{4,5,6,}	{(4,5), (4,6), (5,6)}	{4,4,4}	4	{5,6}	5	{2,3,5}
4	{5,6}	{(5,6)}	{1,1}	1	{6}	5	{2,3,5,6}
5	{}	{ }	--------	-----	------	-	{2,3,4,5}

Ahmad *et al* in [20], proposed a new algorithm for minimum vertex cover, *i.e*. Advanced Vertex Support Algorithm (AVSA). They have carried out small modification in MVSA. Like MVSA, AVSA uses the same data structure introduced in VSA. In AVSA first, the support of each vertex is calculated and the minimum support vertices are found out. All adjacent vertices to minimum support vertices are calculated and the vertex having maximum support of all neighbor vertices of the minimum support vertex is selected as the candidate vertex for MVC and it is removed, this process continues until no edge remains in G. The pseudo code of the AVSA is given in Figure 4.

Advanced Vertex Support Algorithm (AVSA)
Input: G= (V, E).
Output: MVC
Begin:
1. C ←∅;
//initially the vertex cover is empty.
2. while E ≠ϕ do {
3. n ← |G|
4. for i ← 1 to n {
5. d(v_i)
// calculate the degree of each vertex and assign to an array A.
6. SV[i] ← s (v_i)
// calculate the support of each vertex and assign to an array SV.
7. x ← **max**(SV[i])
// calculate the minimum support vertex in G and assign to x.
8. ad_min_supp(x)
// find out all the adjacent vertices of the minimum support vertex.
9. u ← min(ad_min_supp(x))
// find out the minimum support vertex in all adjacent vertices of MSV.
10. V←V- u
11. C ← u
12. Return C.
End

Figure 4. Pseudo Code of Advanced Vertex Support Algorithm (AVSA)

There is no extra complexity involved in computation and decisions is made straightforward. The runtime complexity of the MVSA is O (EV^2log v). The AVSA also fails to provide optimal result on G_1.

Gujral *et al* in [21], introduced a new algorithm called near optimal vertex cover algorithm (NOVCA) which constructs the vertex cover by repeatedly adding, at each step, all vertices adjacent to the vertex having a minimum degree in the graph. In the

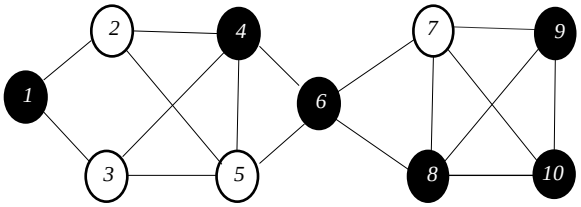
case of a tie, it chooses the one having the maximum sum of degrees of its neighbors. The pseudocode of the NOVCA is given in Figure 5.

```
Near Optimal Vertex Cover Algorithm (NOVCA)
Input: G = (V, E).
Output: MVC
Begin:
1.      C ←∅;
                        // initially vertex cover is empty.
2.      while E ≠ ϕ do {
3.      for i ← 1 to n {
4.      d (vi )
                        //calculate the degree of each vertex of G.
5.      } mind ← (G )
6.      p ← |mindeg|
7.      ad_mindegv ← N (mindeg)
                        //calculate the neighbors of the minimum degree node(s) and assign
to ad_mindeg.
8.      if ( p=1) {
9.      G ← G\{ad_mindegv}
10.     C← C U {ad_mindegv}
11.     If (p>1) {
12.     for i ← 1 to p {
13.     A[i] ← sumd_ad_mindegvi
14.     Go to for
15.     u ← max(A[i])
16.     G ← G\u
17.     }} Go to while
18.     Return C
End
```

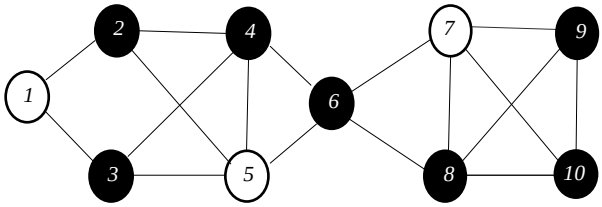
Figure 5. Pseudo Code of Near Optimal Vertex Cover Algorithm (NOVCA)

The working mechanism of NOVCA is given in detail in Table 4 by applying it on G_3. First, the degree of each vertex of G_3 in calculated as shown in row 2, column 5 of Table 4. Then, the minimum degree vertex is calculated, which is '1' having the minimum degree in all vertices of G_3 which is 2 and only vertex in the given graph having the degree 2, and the adjacent vertices of vertex 1 are 2 and 3.

G3



Graph 3. (a). Optimal MVC = {1, 4, 6, 8, 9, 10},



(b). NOACA MVC = {2, 3, 4, 6, 8, 9, 10}

Table 4. Working Mechanism of NOVCA on G3

Sr. No.	V	E	d(V)	mind	T	ad_mind	A[i]	u	C
1	{1,2,3,4,5,6,7,8,9,10}	{(1,2),(1,3),(2,4),(2,5),(3,4),(3,5),(4,5),(4,6),(5,6),(6,7),(6,8),(7,8),(7,9),(7,10),(8,9),(8,10),(9,10)}	{2,3,3,4,4,4,4,3,3}	{1}	1	{2,3}	-----	----	{2,3}
2	{4,5,6,7,8,9,10}	{(4,5),(4,6),(5,6),(6,7),(6,8),(7,8),(7,9),8,9),(7,10),(8,10),(9,10)}	{3,3,4,4,4,3,3,3}	{4,5}	2	{5,6},{4,6}	{6,6}	6	{2,3,4,6}
3	{7,8,9,10}	{(6,8),(7,8),(7,9),8,9),(7,10),(8,10),(9,10)}	{2,2,2,1,1}	{7,8,9,10}	4	{8,9,10},{7,9,10},{7,8,10},{7,8,9}	{9,9,9,9}	9{7}	{2,3,4,6,8,9,10}
4	{}	{ }	------	-------		---------	------	----	{2,3,4,6,8,9,10}

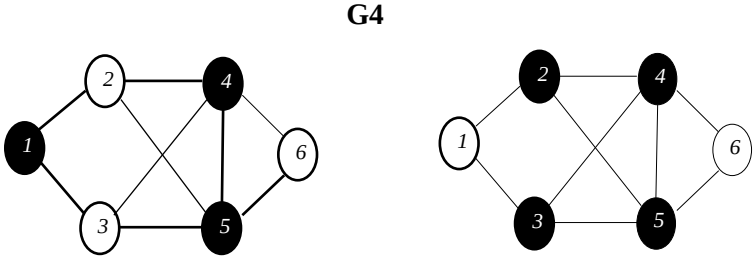
Gujral *et al* in [22], introduced modified version of NOVCA, In the NOVCA-II, all the vertices adjacent to the minimum degree vertex are included in the vertex cover. In the case of the tie that vertex is selected for vertex cover having the minimum sum of degrees of its neighbors. The pseudocode of the NOVCA-II is given in Figure 6.

Near Optimal Vertex Cover Algorithm-II (NOVCA-II)
Input: G= (V, E)
Output: MVC
Begin:
1. C ←∅;
// initially vertex cover is empty.
2. while E ≠ϕ do {
3. for i ← 1 to n {

```
4.      d (vi )
                    //calculate the degree of each vertex of G.
5.      } mindeg ←…. (G )
6.      ad_mindegv ← N (mindeg)
7.      p ← |mindeg|
8.      if ( p=1) {
9.      G ← G\{ad_mindegv}
10.     C← C U {ad_mindegv}
11.     If (p>1){
12.     for i ← 1 to p {
13.     A[i] ← sum_admindegv_i
14.     Go to for
15.     u ← min(A[p])
16.     G ← G\u
17.     }} Go to while
18.     return C
End
```

Figure 6. pSEUDO CODE of Near Optimal Vertex Cover Algorithm-II (NOVCA-II)

Table 5 illustrated that how the NOVCA-II works by applying it on G_4, NOVCA-II has almost same strategy for selection of vertices of MVC as NOVCA.



Graph 4. (a). Optimal MVC = {1, 3, 5} (b). NOACA=II MVC = {2, 3, 4, 5}

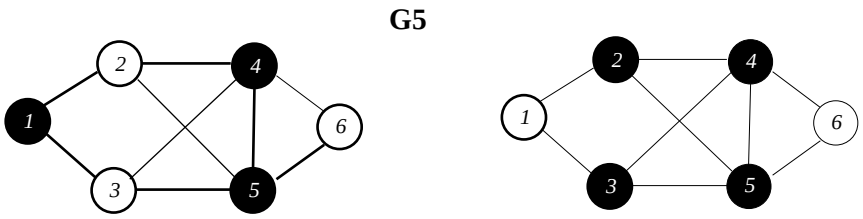
Table 5. The Working Mechanism of NOVCA-II

Sr. No.	V	E	d(V)	mind	p	ad_mind	A[i]	u	C
1	{1,2,3,4,5,6}	{(1,2),(1,3), (2,4), (2,5), (3,4), (3,5),(4,5),(4,6),(5,6)}	[2,3,3,4,4,2]	{1,6}	2	{2,3}, {4,5}	{6,8}	6{1}	{2,3}
2	{,4,5,6}	{(4,5),(4,6),(5,6)}	[2,2,2]	{4,5}	2	{4,5},{4,6},{5,6}	{4,4}	4	{2,3,4,5}
3	{1,6}	{}	-------	------	--	------	------	------	{2,3,4,5}

Li *et al* in [23] introduced a new concept max-share of degree Max-I. According to this concept, the vertex is included in vertex cover which reduced the degree of its neighborhood to zero as much as possible or decreased the value of the degree of all its neighborhood as much as possible. In the case of tie, random selection is made. The pseudo code of the Max-I algorithm is given in Figure 7.


```
Max Share of Degree Algorithm (Max-I)
Input: G = (V, E)
Output: MVC
Begin:
1.      C ← ∅;
                    //initially the vertex cover is empty.
2.      while E ≠ϕ do {
3.      for i ← 1 to n {
4.      d (vi )
                    //calculate the degree of each vertex and assign to an array A.
5.      } u← MSD (vi)
6.      V ← V − {u}.
                    //delete the max share degree vertex from G.
7.      C←C ∪ {u}
                    //add max share degree to vertex cover C.
8.      } return C.
End
```

Figure 7. Pseudo Code of Max Share of Degree Algorithm (MAX-I)



Graph 5. (a). Optimal MVC = {1,4,5}, (b). MAX-I MVC = {2, 3, 4, 5}

Imran *et al*. in [24], proposed a new method, named degree contribution algorithm. In this algorithm first the degree calculation of each vertex is carried in G, next the degree contribution value for each vertex is calculated and the vertex having higher degree contribution value is included in MVC. After the selection of a vertex for MVC, all its adjacent edges are removed. This process continues till no edge remains in the vertex cover set. The pseudocode of the proposed algorithm is given in Figure 8.

```
Degree Contribution Algorithm (DCA)
Input: G = (V, E)
Output: MVC
Begin:
1.      C ← ∅;
                    //initially the vertex cover is empty.
2.      while E ≠ϕ do {
3.      for i ← 1 to n {
4.      d (vi )
                    //calculate the degree of each vertex.
5.      A[i] ←  dc(vi)
                    // calculate the degree contribution of each vertex in G.
6.      u ← max(A)
7.      G ← G\u
8.      }} Go to while
9.      Return C
End
```

Figure 8. Pseudo Code of Degree Contribution Algorithm (DCA)

Halldórsson *et al* in [25], proposed an algorithm based on greedy strategy, The greed is good has been proposed for the Maximum Independent Set (MIS) problem, in this algorithm first the degree of each vertex is calculated by using "Network Bench Model" (Degree Calculation Model) which calculates the degree for each node in O(E) where 'E' is the total number of edges in the graph" and the vertex having the minimum degree of G is selected for MIS. The selected vertex for MIS and all its adjacent vertices are removed from G, this process continues until no edges remain in G. The pseudocode of Greed is Good algorithm is given in Figure 9.

```
Greed is Good (GIG )
Input: G = (V, E)
Output: MIS
Begin:
1.      I ← ∅;
                    //initially the vertex cover is empty.
2.      while E ≠ϕ do {
3.      for i ← 1 to n {
4.      d (vi )
                    //calculate the degree of each vertex.
5.      A[i] ←  d(vi)
6.      u ← min(A)
7.      G ← G-(u & N(u))
8.      }} Go to while
9.      return I
End
```

Figure 9. Pseudo Code of Greed is Good Algorithm

Jovanovic *et al* in [26] proposed ant colony algorithm for MVC, it is based on the heuristic behavior of real ants for searching food. The Ant colony framework is frequently used for optimization problems. The ant colony algorithm was first introduced to solve traveling salesman (TSP) problem, by keeping it in mind that the graph in TSP should by fully connected and there must be weighted edges, but for solving the minimum vertex cover problem the graph is not necessary to be fully connected and here we are dealing with the un-weighted graph, hence, weights on the edges are also not necessary. In this approach the arbitrary edges are established between vertices having no edges among them if an edge is established between two vertices, it is given '0' and if an edge is presented between two vertices in the original graph, then that vertex is weighted '1'. The graph for MVC is set as TSP graph and then the Ant colony is applied on the graph for calculating MVC vertices. In this paper, an idea is given but no details of implementation and results are provided. It is still very challenging problem to get MVC for un-weighted graph by applying ant colony algorithm.

3. Comparative Analysis

In Table 6 the different comparative analysis of some state of art approximations algorithms for MVC has been carried, The comparative study of these approaches will help us to understand the weakness and strengthens of these approximation algorithms for MVC and in the light of theses we would be able to design a simple, fast and efficient approximation algorithm for MVC. The run time complexity of each of algorithm which is described in the Table 6 given bellow.

Table 6. Approximation Algorithms for MVC Problem with Their Properties

Ref. No.	Algorithm	Properties
[5]	Vertex Support Algorithm (VSA)	• It takes O (En^2) to compute. • It required sound efforts to implement. • A vertex selection for MVC is little bit complex. • It has 91% success rate. • It provides better approximation ratio.
[17]	Maximum Degree Algorithm (MDG)	• It takes O (V^2) to compute. • Easy to program and easy to understand. • It has 56% worst success rate. • Criteria for selection of vertex for MVC is very simple. • It is best for solving graphs having no cycle.
[19]	Modified Vertex Support Algorithm (MVSA)	• It is the modified version of VSA. • The same data structure is used as in VSA. • Complex in implementation as compared to VSA. • It provides better approximation ratio as compared to VSA.
[20]	Advanced Vertex Support Algorithm (AVSA)	• Like MVSA, it is also the modified version of VSA. • Data structure and selection criteria of a vertex for MVC is simple. • Same complex as MVSA.
[21]	Near Optimal Vertex Cover Algorithm (NOVCA)	• Its run time complexity is O ($n^2 \log n$). • Easy to implement and easy to understand. • It worst rate is 55%. • Criteria for selection of a vertex for MVC is simple.
[20]	Near Optimal Vertex Cover Algorithm-II (NOVCA-II)	• Runtime complexity is = O (V^2 ($V \log^2 V$)). • Like NOVCA, it is simple to implement and easy to understand. • The worst success rate is 50%. • Simple criteria for vertex selection in vertex cover set. • Almost same idea as NOVCA-I but has slight change. • It is not the modified version of NOVAC.
[23]	Max Share of Degree Algorithm (Max-I)	• The time it takes to compute is O (n^2). • Somehow prediction is involved and its implementation is a challenging task. • Worst approximation ratio of it is 1.01 when applying it on large benchmarks graphs. • Can give worst result when average results increase than 2.
[24]	Degree Contribution Algorithm (DCA)	• It very fast even from the simplest algorithm of MVC MDG. • Its implementation is very simple and vertex selection for MVC criteria is very simple. • It provides better approximation on average benchmarks graphs.
[25]	Greed is Good (GIG)	• It takes O($V^{2)}$ to compute. • It is the simplest algorithm for MIS. • Its worst success rate is 75%. • It gives optimal results on trees.
[26]	Ant Colony Algorithm for MVC	• It takes O (n^2) for computation. • Just the idea is given no experiments and implementation details. • It is difficult to implement ant colony algorithm for MVC. • It is not suitable for MVC problem.

4. Conclusion

Different type of pseudo code and algorithms are used for representing the NP problems approaches, Most of the authors used different symbols in their pseudo codes which actually make it complex for the reader to easily understand the proposed approaches for a problem, hence there was need of such literature server which uses same type of symbols of presentation for each pseudo code in order to easily the algorithms understandable for readers, hence in the proposed approach we have carried a detail literature review for this purpose for minimum vertex cover problem.

The NP problems is consider difficult area for study because when one start study on a certain problem so first the researchers face difficulties to understand different algorithm for that problem and second challenge that on which instances the given algorithm fails to provide optimal results, hence here we have also provided the small benchmarks on which the given algorithm fails to provide optimal result, which make this literature useful for researcher to develop a simple and efficient approximation for minimum vertex cover problem.

References

[1] D. B. West, "Introduction to graph theory", vol. 2: Prentice hall Upper Saddle River, **(2001)**.
[2] M. Fayaz, S.Arshad, A.S.Shah and A.Shah, "Max Degree Around (MDA) Algorithm: A Smart and Efficient Approximate Algorithm for Vertex Cover and Independent Set Problems", Sindh University Research Journal-SURJ (Science Series), vol. 48, no. 4D, **(2016)**, pp. 17-26.
[3] M. Fayaz, S. Arshad, A.S. Shah and A. Shah, "An Optimal Approximation Algorithm for Optimization of Un-Weighted Minimum Vertex Cover Problem", Sindh University Research Journal-SURJ (Science Series), vol. 48, no. (4D), **(2016)**, pp. 175-182.
[4] S. Richter, M. Helmert and C. Gretton, "A Stochastic Local Search Approach to Vertex Cover", In: Hertzberg J., Beetz M., Englert R. (eds) KI 2007: Advances in Artificial Intelligence. KI 2007. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, vol 4667, **(2007)**, pp. 412-426.
[5] S. Balaji, V. Swaminathan and K. Kannan, "Optimization of Unweighted Minimum Vertex Cover", World Academy of Science, Engineering and Technology, vol. 67, **(2010)**, pp. 508-513.
[6] M. Weigt and A. K. Hartmann, "Number of Guards Needed by a Museum: A Phase Transition in Vertex Covering of Random Graphs", Physical review letters, vol. 84, no. 26, **(2000)**, p. 6118.
[7] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", W. H. Freeman & Co. New York, NY, USA, **(1979)**.
[8] M. Fayaz, and S. Arshad, "Clever Steady Strategy Algorithm: A Simple and Efficient Approximation Algorithm for Minimum Vertex Cover Problem", In Proceedings of the 2015 13th International Conference on Frontiers of Information Technology, IEEE, 14-15 December, **(2015)**.
[9] R.M. Karp, "Reducibility among combinatorial problems", R.E. Miller and J.W. Thatcher eds., Complexity of Computer Computations, 85-103, Plenum Press, NY, **(1972)**.
[10] G.J. Woeginger, "Exact algorithms for NP-hard problems: A survey", In: Combinatorial Optimization—Eureka, You Shrink. Lecture Notes in Computer Science, vol. 2570, **(2003)**, pp. 185–207. Springer, Berlin.
[11] V. Kann, "On the Approximability of NP-complete Optimization Problems", Ph.D. Thesis. Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, **(1992)**.
[12] V. V. Vazirani and V. Vijay, "Approximation algorithms", Springer-Verlag Berlin Heidelberg, **(2003)**.
[13] G. Karakostas, "A better approximation ratio for the vertex cover problem", In: Proceedings of the 32nd International Colloquium on Automata, Languages and Programming, ICALP, Lisboa, Portugal, July **(2005)**, pp. 1043–1050.
[14] M. Fayaz, S. Arshad, U. Zaman and A. Ahmad, "A Simple, Fast and Near Optimal Approximation Algorithm for Optimization of Un-weighted Minimum Vertex Cover", In Frontiers of Information Technology (FIT), 2016 International Conference on, IEEE, **(2016)**, pp. 176-180.
[15] A.I. Bykov, and L.G. Zerkalov, "Algorithm for Homotopy Classification of Binary Images", Pattern Recognition, vol. 29, no.4, **(1996)**, pp. 565-574.
[16] Balaji. S, V. Swaminathan and K. Kannan, "A Simple Algorithm to Optimize Maximum Independent Set", Advanced Modeling and Optimization, vol. 12, no.1, **(2010)**, pp.107-118.
[17] K. L. Clarkson, "A Modification of the Greedy Algorithm for Vertex Cover", Information Processing Letters, vol. 16, **(1983)**, pp. 23-25.
[18] V. Chvatal, "A Greedy Heuristic for the Set-Covering Problem", Mathematics of Operations Research, vol. 4, **(1979)**, pp. 233-235.

[19] K. Imran and K. Hasham, "Modified Vertex Support Algorithm: A New approach for approximation of Minimum vertex cover", Research Journal of Computer and Information Technology Sciences, vol.1.no.6, **(2013)**, pp. 7-11.
[20] I. Ahmad and M. Khan, "AVSA, Modified Vertex Support Algorithm for Approximation of MVC", International Journal of Advanced Science and Technology, vol. 67, **(2014)**, pp. 71-78.
[21] S. Gajurel and R. Bielefeld, "A Simple NOVCA: Near Optimal Vertex Cover Algorithm", Procedia Computer Science, vol. 9, **(2012)**, pp. 747-753.
[22] S. Gajurel and R. Bielefeld, "A Fast Near Optimal Vertex Cover Algorithm (NOVCA)", IJEA, vol. 3, no.1, **(2012)**, pp. 9-18.
[23] S. Li, J. Wang, J. Chen and Z. Wang, "An Algorithm for Minimum Vertex Cover Based on Max-I Share Degree", Journal of Computers, vol. 6, no.8, **(2011)**, pp. 1781-1788.
[24] I. khan and H. khan, "Degree Contribution Algorithm for Approximation of MVC", International Journal of Hybrid Information Technology, vol. 7, no.5, **(2014)**, pp. 183-190.
[25] M. M. Halldorsson and J. Radhakrishnan, "Greed is good: Approximating independent sets in sparse and bounded-degree graphs," Algorithmica, vol. 18, no.1, **(1997)**, pp. 145-163.
[26] R. Jovanovic and M. Tuba, "An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem", Applied Soft Computing, vol. 11, no. 8, **(2011),** pp. 5360-5366.

Authors

Muhammad Fayaz, he is currently perusing Ph.D. in Computer Science. He received MS in Computer Science from SZABIST, Islamabad, Pakistan in 2014. He did MSC from the University of Malakand, KPK, Pakistan in 2011. His areas of interest are NP problems, Approximation Algorithms, Image Processing, and Pattern Recognition.

Shakeel Arshad, he is Assistant Professor at university of Malakand, Dir(L),KPK, Pakistan, His research interest area are Genetic Algorithm and Machine learning, theoretical computer science, and more specifically in approximation algorithms, computational geometry, distributed algorithms, and randomization.

Abdul Salam Shah, he is currently perusing Doctor of Philosophy (Information Technology) at Malaysian Institute of Information Technology, University of Kula-Lumpur (UniKL), Kula-Lumpur, Malaysia. Furthermore, he has completed MS degree in Computer Science from SZABIST, Islamabad, Pakistan in 2016. He did his BS degree in Computer Science from Isra University Hyderabad, Sindh Pakistan in 2012. In addition to his degree, he has completed short courses and diploma certificates in Databases, Machine Learning, Artificial Intelligence, Cybercrime, Cybersecurity, Networking, and Software Engineering.

He has completed specialization in Management Information System (MIS) from Virtual University of Pakistan in 2017. He has published articles in various journals of high repute. His research area includes Machine Learning, Artificial Intelligence, Digital Image Processing, Cybersecurity, and Data Mining.

Asadullah Shah, he is working as Professor and Head of department of Information Systems (HOD) at the Kulliyyah of ICT, International Islamic University Malaysia (IIUM) before joining IIUM, he worked as Head of Telecommunication Engineering & Management department, IoBM Karachi Sindh, Dean Faculty of Computer and Management Sciences, Isra University Hyderabad Sindh and Head of Telecommunication Engineering and IT, Sukkur IBA, Sindh-Pakistan.

He did his Ph.D. from the university of Surrey UK, in 1998, with the specialization in Multimedia Communication. He started his academic carrier from University of Sindh Jamshoro, Pakistan in 1986 as a lecturer.

He has published 200 research articles in highly reputable international and national journal in the field of computers, communication and IT. Also, he has published 12 books in his 30 years of the academic carrier. Currently he is supervising great number of postgraduate students, working in multiple disciplines, specially, animation, social media and image processing in the Department of Information Systems, Kulliyyah of Information and Communication Technology, International Islamic University Malaysia.

