A Simple and Efficient Algorithm for Optimization of Minimum Vertex Cover Problem

Aslam Khan and Said Khalid Shah

Department of Computer Science, University of Charsada, Charsada, Pakistan
Department of Computer Science, University of Science & Technology, Bannu, Pakistan

Abstract
This paper represent a very simple and efficient Polynomial time and Maximum degree contribution algorithm (MDCA). The proposed algorithm consists of three parts, first the degree of each vertex is calculated and it is checked as there a vertex having degree equal to 1, if yes then the neighbor of that vertex is added to MVC. If more than one vertex having degree equal to '1' then randomly one of them is selected and the neighbor of that vertex is added to MVC. In second step, If G= (V, E) have no vertex having degree equal to 1 then the vertex having maximum degree in the given graph (G) and its neighbors are determined. The edges are removed one by one of the maximum degree vertex, first that vertex is removed which is attached to the minimum degree vertex, if the degree of that minimum degree vertex becomes '1'then its neighbor is added to MVC. The edges between maximum degree vertex and its neighbors are removed one by one by starting from the minimum degree neighbor vertex and after removing all edges and the degree of maximum degree vertex become '0' then the maximum degree vertex itself is added to MVC. This whole process continues until the graph becomes empty. This algorithm is tested on small as well as on large bench mark instances. The experimental results and the comparative analysis show that MDCA gives better and fast solution as compare to others approximation algorithms found in the literature for solving minimum vertex cover problem.

Keywords
Minimum vertex cover (MVC); maximum independent set (MIS); NP –complete problem; optimization problem; approximation algorithm; maximum degree greedy (MDG); advanced Vertex Support Algorithm(AVSA);Modified Vertex Support Algorithm(MVSA); and maximum degree contribution algorithm(MDCA)

1. Introduction

A graph is said to be the collection of two or more than two nodes which are attached to each other's through edges. A graph can be represented mathematically as, G= ("V, E") where 'V' stands for vertices and 'E' for edges. Graphs have an important role to model i.e. represent and simulate real world problems such as social and information systems in general and computer communication networks in specific. Computer network of communications, management of data, devices computation , the data flow , etc. can be represented by using graph[1, 2].

The problem of minimum vertex cover called as (MVC) is the smallest set of nodes or vertices in G= (V, E) which is used to manage all links in a graph. In the problem of MVC we have to select smallest no of nodes to manage all the links in the given Graph.

The graph can be covered by using many set of vertices, but in MVC smallest set of vertices are required to cover all the links of the graph. It is an optimization problem because in MVC we have to find the best solution. It is a challenging problem due to its property because in this problem we have to choose the smallest number of nodes to manage all the links in a graph.

Here in this section we are going to study the related problems and terminologies with minimum vertex cover problem in de Computation problems can be divided into three different classes, Polynomial problems, non-deterministic Polynomial problems (NP-Problems) and NP-Complete problems.

A problem is said to be the polynomial problem if it can be calculated in polynomial time to acquire the desire solution are called polynomial problems, such as complement of a graph, subtraction of one graph from another etc., problems of minimum vertex cover can be calculated in polynomial time when graph size is small[3].

A problem which can't be calculated in 'polynomial time' is called NP-problem, it takes exponential time to compute when the problem size is large, example of NP-problems are traveling sales For the problem to be NP complete it should possessed two properties. The first one is that it should be NP problem complete and second if it is not NP complete problem than it should be reduced to NP complete problem in polynomial time. [4]. man, Hamiltonian cycle, graph coloring problems[4], etc. There are two algorithms which is used to solve the NP-complete problems, approximation and complete algorithms. One way to solve the NP-complete problems is complete algorithms. It always give exact solution of problem of NP but the drawback of this algorithm is that the time taken by this algorithm is increased exponentially with the extension of the size of problem. Even for solving a normal size of problem these can takes one million years or in other words a trillion years by the use of current computation power [4-6]. The exact or complete solution of a NP problem is

Manuscript received May 5, 2026
Manuscript revised May 20, 2026
https://doi.org/10.22937/IJCSNS.2026.26.5.11

applicable where size of problem is very small and complete solution is needed without any time constrain[6].

The approximation algorithms are very popular algorithms and it always provides the best solution of the problem within the given time frame. Approximation algorithms are best choice with approximate solution is required in quick time of a large benchmarks graphs [7-9].

The mathematical representation of approximation ratio is as in equation 1 for MVC.

$$\rho i = \frac{Ai}{Opti} \geq 1 \qquad (1)$$

Where Ai shows the results return by these approximation algorithm and the Opti represent the optimal solution of a graph. The value of ρi must be always greater or equal to 1 ($\rho i \geq 1$), $\rho i = 1$ shows the solution given by an approximation algorithm is optimal i.e. best one. More deviation from '1' indicates poor solution of an approximation algorithm[10].

2. Literature Review

The survey of some well-known approximation algorithm is briefly described in this section. These algorithms are chosen on the basis of simplicity, optimality and run time complexity.

A. Maximum degree algorithm(MDG)

The maximum degree Greedy (MDG) is the first simplest and fastest algorithm which we have presented in the literature review[18] . It is based on greedy approach. It was devised by Chvatal in 1979 [19]. It is a very simple algorithm for attempting the VC problems. It selects any node in arbitrary order; add it to MVC after deleting all the attached edges. In this algorithm that vertex which has the maximum degree in the given graph has to chosen and it is added to vertex cover set. After that all the adjacent links of that vertex is deleted one by one. This process continues until the graph become empty. Its drawback is that it even fails on small benchmarks instances .

B. Vertex Support Algorithm (VSA)

The second type of approximation algorithm in this list is vertex support algorithm (VSA) .Vertex support can be define as "the summation of all the degrees of neighbor nodes of a node" In this algorithm first of all the support of each node is calculated and then a vertex having 'maximum support' values among all these nodes are added to MVC. The worst running time complexity of this algorithm is as that of [5]. Just like greedy algorithm this algorithm also fails on some small bench mark instances as well.

C. Modified vertex support algorithm (MVSA)

The modified vertex algorithm (MVSA) also depends upon the same data structure as that of algorithm of VSA. According to this algorithm the degree of each node is first calculated, after that the vertex support of each node is also determined. Then the minimum support node is found out among these nodes. And finally all the neighbor nodes of support minimum are find out, and that node which has support minimum value in these neighbors is added to MVC. In this algorithm there is no extra complexity involved in computation. Here the decision is made very straight forwarded. Like MDG and VSA it also give some poor results on some small bench marks instances.

3. Proposed Algorithm

The competing algorithms as mention in literature review are not very efficient and do not produce optimal results as well and hence having many deficiencies. The MDG algorithm is simple and fast but not optimal. Similarly, VSA algorithm is not much fast and optimal. The MVSA is optimal but not fast as MDG. All of these algorithms have some limitations and there is a need of an algorithm to overcome the limitations of these algorithms. For this purpose, we have designed a type of algorithm which is as fast and simple as MDG and optimal as MVSA. The proposed algorithm up to some level based on greedy approach, but some tweaks have been used intelligently to keep it simple and fast as greedy and optimal as other heuristic approbation algorithms for MVC.

The proposed algorithm consists of three parts, first the degree of each vertex is calculated and it is checked as there a vertex havening degree equal to 1, if yes then the neighbor of that vertex is added to MVC. If more than one vertex having degree equal to '1' then randomly one of them is selected and the neighbor of that vertex is added to MVC. In second step, If G= (V, E) have no vertex having degree equal to 1 then the vertex having maximum degree in the given graph (G) and its neighbors are determined. The edges are removed one by one of the maximum degree vertex, first that vertex is removed which is attached to the minimum degree vertex, if the degree of that minimum degree vertex becomes '1'then its neighbor is added to MVC. The edges between maximum degree vertex and its neighbors are removed one by one by starting from the minimum degree neighbor vertex and after removing all edges and the degree of maximum degree vertex become '0' then the maximum degree vertex itself is added to MVC. This whole process continue until the graph becomes empty.

D. Data structure used in our proposed algorithm

Several data structures have been used graph representation. One way is 'adjacency matrix' which is an n-by-n matrix, i.e. 'matrix of zeroes and ones', where M[i][j] is 1 if and only if the edge (i, j) is in E.

It requires O (n2) space. Another graph data structure is simply an edge list which is a set of attach edges, it needs O(m) space and can be shown with the help of set of pairs. In the proposed approach the edge list data structure has been used [18].

E. Terminologies used in our proposed algorithm

In this section some terminologies which has been used are discussed briefly here. Let G= (V, E) be a graph where V = {v1, v2, v3… vn} is the set of vertices and E= {e1, e2, e3 ……en}, the set of edges. Generally the number of nodes in a graph is represented by n = 'V'|, and the number of edges is represented by m = 'E'. Degree of a vertex is represented by d ( ). It is used to calculate the degree of a vertex, neighbors of a node is represented by N ('v') = { 'u∈ V /u' is adjacent to v} where v ∈ V, | | is used to return number of vertices, Δ and δ symbols are used to represent maximum degree and minimum degree in G= (V, E) respectively. In the Pseudo Code of MDCA algorithm the text to the right side of // represents comments for better elaboration of mathematical notations used in the proposed approach.

F. Pseudo code of the proposed Algorithm (MDCA)

Following is the pseudo code of our proposed algorithm MDCA; input of this algorithm is a graph and out is MVC.

Input to algorithm = ("V, E")
Output of algorithm ='MVC'
Start:
1. C =0;
// in the beginning C is empty
2. While E ≠ ϕ do {
3. n← |V|
// calculate number of nodes in Graph and assigning to n
4. for i ←1 to n {
5. Calculate d (vi) }
// calculate degree of each vertex
6. md← δ (G)
// find out minimum degree in G and assign to md
7. if (md =1) {
// if minimum degree in G is equal to 1
8. u ←N(md)
//assign neighbor of minimum degree vertex to u
9. Go to 23
10. If (md >1)
// if the degree of minimum degree vertex in G is greater than 1
11. x ← Δ (G)
// find maximum degree vertex in G and assign to x
12. Calculate N (x)
// calculate neighbors vertex 'x'
13. p ← |N(x)|
// calculate the number of neighbors of x and assign to p
14. for j← 1 to p {
15. Remove ej∈x and Nj(x)
16. If (d(Nj(x))=1){
// If the degree of neighbor vertex of x is equal to 1
17. u ← N(Nj(x))
// Assign neighbor of that neighbor of x to u
18. Go to 23}
19. if (d(x) = 0) {
// If the degree of vertex x becomes 0 after deleting edges one by one
20. u ← x
// Assign vertex x to u
21. Go to 23}}
22. G ← G\u
23. C←C U u
24. } Go to while
25. Return C
End

4. Implementation, Empirical Results and Discussion

In order to better analyze and understand the theoretical results of our 'MDCA' algorithm along with other competing algorithms i.e. as MDG, VSA and MVSA, we have done our coding in MATLAB R2010a version 7.10.0.499 running window 7 based Intel core i5 systems. We tested our proposed algorithm and the others competing algorithms by applying them on small benchmarks instances first. These small benchmarks instances were used as an input to all the algorithms. This way of running an algorithm on small benchmarks instances will give us a clear picture of the weaknesses and strengthens of the proposed algorithm and as well as the competing algorithms. After pointing the weaknesses and strengthens of these algorithms will be helpful to design a better algorithm for MVC. We compared the results of our proposed algorithm with some well-known approximation (competing) algorithms are shown in Table 1.

Further, we also tested our proposed algorithm on large benchmarks instances as well such as on BHOSLIB and DIMACS graphs. Testing an algorithm on large benchmarks illustrated the overall performance of an algorithm such as time complexity and its optimality.

It is demonstrated here that how our propose algorithm performs for large instances of DIMACS[29] and BHOSLIB [30]benchmarks. Four algorithms are compared in the performance test. In table 6the MDCA approximation algorithm have been compared with some famous approximation algorithm for MVC, the approximation ratios have also been calculated to better exhibit the performance of each algorithm .In table 7the worst and average approximation of each algorithm has been listed. The worst approximation ratios of an algorithm show the poorest performance on certain number of graphs while on the other

Table 1 Performance of 'MDCA', 'MDG', 'VSA' and 'MVSA' on various graphs

'Benchmarks'	Vertices	Optimal results	'MDCA' Algrtm	'MDG' algrtm	'VSA' algrtm	'MVSA' algrtm	'MDCA' ρ	'MDG' P	'VSA' ρ	'MVSA' ρ
graph50-6	50	38	38	38	44	38	1	1	1.159	1
graph50-10	50	35	35	35	41	35	1	1	1.172	1
graph100-1	100	60	60	60	95	60	1	1	1.584	1
graph100-10	100	70	70	70	96	70	1	1	1.372	1
graph200-5	200	150	150	150	184	150	1	1	1.228	1
graph500-1	500	350	350	350	485	350	1	1	1.387	1
graph500-2	500	400	400	400	484	400	1	1	1.211	1
graph500-5	500	290	290	290	454	290	1	1	1.566	1
phat300-1	300	292	292	293	292	294	1	1.003	1	1.008
phat300-2	300	275	275	278	275	279	1	1.010	1	1.016
phat300-3	300	264	266	269	264	272	1.006	1.019	1	1.031
phat700-1	700	689	692	693	689	692	1.003	1.006	1	1.005
phat700-2	700	656	658	660	656	660	1.003	1.007	1	1.007
phat700-3	700	638	642	642	638	649	1.005	1.007	1	1.018
johnson8-2-4	28	24	24	24	24	24	1	1	1	1
johnson8-4-4	70	56	56	62	56	56	1	1.108	1	1
johnson16-2-4	120	112	112	112	112	112	1	1	1	1
johnson32-2-4	496	480	480	480	480	480	1	1	1	1
sanr200-0.7	200	182	183	184	182	186	1.0053	1.011	1	1.022
sanr200-0.9	200	158	162	164	158	163	1.0252	1.038	1	1.032
sanr400-0.5	400	387	388	392	387	389	1.0024	1.013	1	1.006
sanr400-0.7	400	379	382	384	379	381	1.006	1.014	1	1.006
fbr35-17-2	595	560	565	570	573	565	1.008	1.018	1.023	1.010
Fbr-30-15-5	450	420	425	429	429	424	1.010	1.021	1.022	1.010
c-125	125	91	94	93	91	95	1.031	1.022	1	1.044
C-250.9	250	206	209	211	206	211	1.0144	1.024	1	1.025
C-500.9	500	443	452	453	443	449	1.019	1.023	1	1.014
C-2000.9	2000	1922	1925	1944	1923	-	1.001	1.012	1.002	----
brock200-1	200	188	190	190	188	191	1.009	1.011	1	1.015
brock200-4	200	183	192	192	183	193	1.048	1.050	1	1.054
hamming6-2	64	32	32	32	32	32	1	1	1	1
hamming6-4	64	60	60	60	60	60	1	1	1	1
hamming8-2	256	128	128	128	128	128	1	1	1	1
hamming8-4	256	240	240	240	240	240	1	1	1	1
hamming10-2	1024	512	512	512	512	512	1	1	1	1
dsjc-500	500	487	489	491	487	489	1.003	1.009	1	1.005
Killer-4	171	160	160	164	160	160	1	1.026	1	1
Killer-5	776	749	754	764	749	754	1.005	1.021	1	1.007
c-fat200-1	200	188	188	188	188	188	1	1	1	1
c-fat200-2	200	176	176	176	176	176	1	1	1	1
c-fat200-5	200	142	142	142	144	142	1	1	1.015	1
c-fat500-1	500	486	486	486	486	486	1	1	1	1
c-fat500-2	500	474	474	474	474	474	1	1	1	1
c-fat500-5	500	436	436	436	436	436	1	1	1	1
c-fat500-10	500	374	374	374	374	374	1	1	1	1
MANN-a27.clq.b	378	252	253	261	253	253	1.003	1.036	1.005	1. 005

hand the average approx ratios of an algorithm show the performance of all graphs. In table no 3 the 'worst' and 'average' approx ratios have less deviation than the rest of competing algorithms and hence the performance of the proposed algorithm is better than the other competing algorithms. In table 8 the time taken by each algorithm is carried out, which shows that our MDCA algorithm is efficient when it is compared with others, competing algorithms.

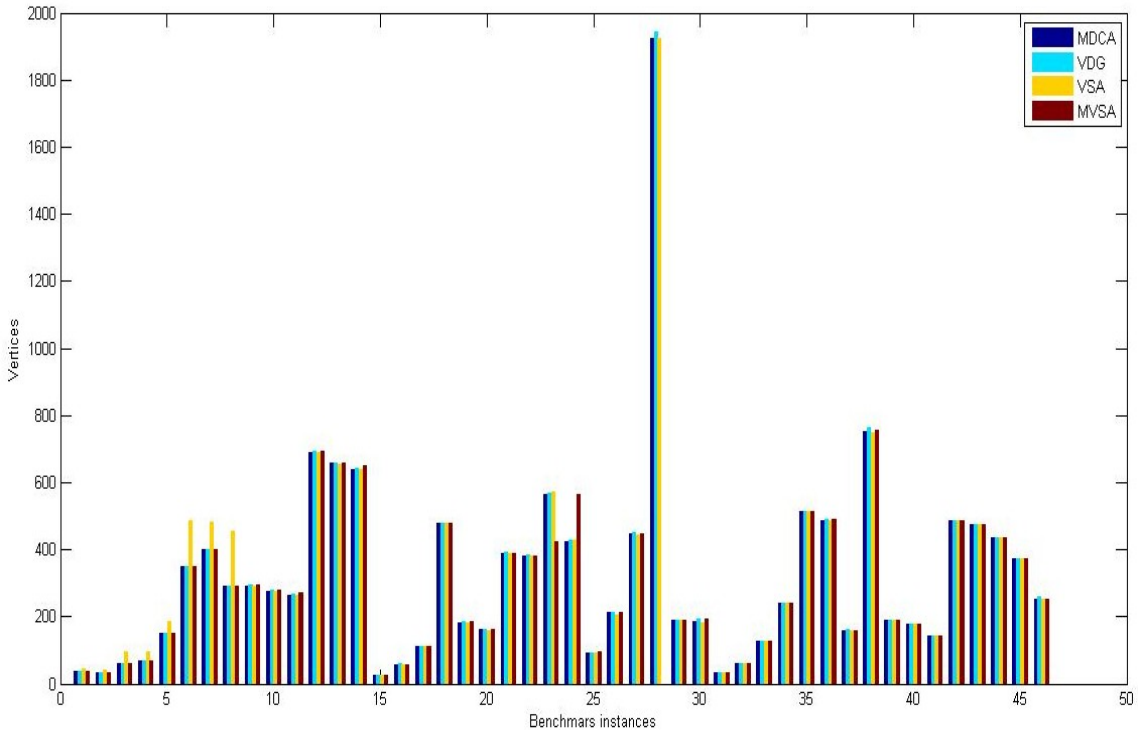

Figure 1 graphical representation of table 1.

G. Worst and average approximation ratios:

In below table 2 the worst and average approximation ratios are listed for each of the given algorithms. If and algorithm has high worst ratios it demonstrates that on specific bench mark instances the given algorithm yields very poor result. An average approximation ratios exhibit that the performance of the algorithm on all bench mark instances is not satisfactory. The worst and average approximation ratios of the proposed algorithm are less than the counterpart's algorithms which indicates the efficiency of our proposed algorithm with respect to optimality.

Table 2 Worst and Average Approximation Ratio of MDCA, MDG, VSA, AND MVSA

'Algorithms'	**'Worst'** ρ_i	**'Average'** ρ_i
'MDCA'	1.0252	1.002963
'MDG'	1.108	1.012614
'VSA'	1.584	1.05946
'MVSA'	1.065	1.009

H. Comparison of time taken by an algorithm:

In below table 3 we compared the MDCA with MDG, VSA and MVSA, w.r.t time (sec) . In this section the proposed algorithm is compared with the simplest and fast algorithms to show that how our proposed algorithm MDCA runs fast on individual benchmarks instances.

5. Conclusion

In this paper we have carried out in detail literature study of some well-known approximation algorithm for minimum vertex cover (MVC)problem. All these algorithms have been selected because these algorithms are very simple, fast and optimal having low running time. Our proposed algorithm was fist tested on small bench mark instances which beaten the counterpart algorithms. After that the proposed algorithm was also tested on large bench marks instances such as DIMACS and BHSLIB and the result showed that MDCA is fast and efficient optimization algorithm as compared to MDG, VSA and MVSA. Our proposed algorithm is better in real application where a better approximation results are required in short span of time. Our algorithm is also simple and easy to implement as compared to others algorithm.

Table 3 Performance of Benchmarks on various graphs

'Benchmarks'	***Vertices***	***'MDCA'time***	***'MDG' time***	***'VSA'time***	***'MVSA' time***
graph50-6	50	0.125	0.1093	0.2032	0.4355
graph50-10	50	0.1563	0.0937	0.1876	0.3320
graph100-1	100	1.2031	0.6	1.2032	1.6755
graph100-10	100	1.3281	0.6406	1.2657	1.9212
graph200-5	200	4.4219	4.9532	9.8595	12.2533
graph500-1	500	78.4063	168.24	363.1876	400.238
graph500-2	500	173.6406	540.1093	643.5157	703.326
graph500_5	500	141.1250	351.3280	738.2970	845.3215
Hamming6-2	64	0.0938	0.0626	0.2032	0.553
Hamming6-4	64	0.3906	0.2189	0.3439	0.562
Hamming8_2	256	1.3125	0.9376	2.376	4.31
Hamming8-4	256	12.7031	17.7655	30.8907	40.234
Hamming10-2	1024	29.1719	53.5625	52.3282	60.31
Hamming10-4	1024	3050	3030	3171	3201
jhonson8-4-4	70	0.1719	0.1876	0.2814	0.5320
jhonson8-2-4	28	0.0469	0.0314	0.0626	0.1244
jhonson32-2-4	496	34.0469	95.4377	1161	1359
c-125	125	0.4063	0.3594	0.626	0.9353
C-250	250	3.4844	2.6564	4.26	7.234
Sanr-200-0.7	200	3.5938	3.7187	6.7345	11.230
Sanr-200-0.9	200	1.6875	1.4219	2.4220	3.2154
Sanr-400_0.5	400	46.4375	93.3437	161.1720	171.324
sanr400_0.7	400	30.7969	50.8282	94.2189	199.324
gen200-p0.9-44	200	2.5839	1.4064	2.6095	3.214
Cfat-200-1	200	6.9844	11.8439	21.0782	26.323
Cfat-200-2	200	6.3438	10.2968	18.7814	25.212
Cfat-200-5	200	5.1406	5.4843	8.4220	11.214
Cfat-500-1	500	340.9375	1059	1491	1511
c-fat500-2	500	333.2656	977.1563	1381	1449
c-fat500-5	500	223.9065	611.8124	786.4064	936.253
Phat-300-1 c	300	25.7969	38.8280	59.8907	70.324
Phat-700-1 c	700	973.5156	2551	2401	2599
Phat-700-2 c	700	657.4531	1859	2461	2601
Mann-a27 c	378	3.3750	2.0626	3.9845	6.324
Dsjc-500	500	108.6719	256.875	277.994	333.264
Keller-4 c	171	2.6094	2.6405	4.2814	10.320
Keller-5 c	776	288.4375	894.6718	1228	1344

References

[1] D. B. West, *Introduction to graph theory* vol. 2: Prentice hall Upper Saddle River, 2001.

[2] J. A. Bondy and U. S. R. Murty, *Graph theory with applications* vol. 290: Macmillan London, 1976.

[3] R. E. Ladner, "On the structure of polynomial time reducibility," *Journal of the ACM (JACM),* vol. 22, pp. 155-171, 1975.

[4] M. R. Garey and D. S. Johnson, "A Guide to the Theory of NP-Completeness," *San Francisco,* 1979.

[5] R. M. Karp, *Reducibility among combinatorial problems*: Springer, 1972.

[6] G. J. Woeginger, "Exact algorithms for NP-hard problems: A survey," in *Combinatorial Optimizationâ€"Eureka, You Shrink!*: Springer, 2003, pp. 185-207.

[7] D. S. Hochbaum, *Approximation algorithms for NP-hard problems*: PWS Publishing Co., 1996.

[8] G. Ausiello, *Complexity and approximation: Combinatorial optimization problems and their approximability properties*: Springer, 1999.

[9] V. V. Vazirani, *Approximation algorithms*: Springer Science & Business Media.

[10] G. Karakostas, "A better approximation ratio for the vertex cover problem," in *Automata, languages and programming*: Springer, 2005, pp. 1043-1050.

[11] M. C. Golumbic, *Algorithmic graph theory and perfect graphs* vol. 57: Elsevier, 2004.

[12] S. Boyd and L. Vandenberghe, *Convex optimization*: Cambridge university press, 2004.

[13] 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, p. 6118, 2000.

[14] S. Balaji, V. Swaminathan, and K. Kannan, "Optimization of unweighted minimum vertex cover," *World Academy of Science, Engineering and Technology,* vol. 43, pp. 716-729.

[15] D. Kumlander, "Comparing the best maximum clique finding algorithms, which are using heuristic vertex colouring," in *Proceedings on the 10th WSEAS International Conference on Computers*, 2006, pp. 932-937.

[16] K. Baamann, "The Maximum Clique Problem-On Finding an Upper Bound with Application to Protein Structure Alignment," Georgia Institute of Technology, 2003.

[17] S. Pollatos, "Solving the maximum clique problems on a class of network graphs, with applications to social networks," Monterey, California. Naval Postgraduate School, 2008.

[18] K. L. Clarkson, "A modification of the greedy algorithm for vertex cover," *Information Processing Letters,* vol. 16, pp. 23-25, 1983.

[19] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of operations research,* vol. 4, pp. 233-235, 1979.

[20] 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, ISSN,* vol. 2320, p. 6527.

[21] I. Ahmad and M. Khan, "AVSA, Modified Vertex Support Algorithm for Approximation of MVC," *International Journal of Advanced Science and Technology,* vol. 67, pp. 71-78.

[22] S. Gajurel and R. Bielefeld, "A Simple NOVCA: Near Optimal Vertex Cover Algorithm," *Procedia* [18] M. Fayaz and S. Arshad, “Clever Steady Strategy Algorithm: A Simple and Efficient Approximation Algorithm

for Minimum Vertex Cover Problem," in 2015 13th International Conference on Frontiers of Information Technology (FIT) , pp.277-282.

[23] S. Gajurel and R. Bielefeld, "A fast near optimal vertex cover algorithm (novca)," *IJEA,* vol. 3, pp. 9-18.

[24] 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, pp. 1781-1788.

[25] I. khan and H. khan, "Degree Contribution Algorithm for Approximation of MVC," *International Journal of Hybrid Information Technology,* vol. 7, pp. 183-190.

[26] M. s. M. HalldÃ³rsson and J. Radhakrishnan, "Greed is good: Approximating independent sets in sparse and bounded-degree graphs," *Algorithmica,* vol. 18, pp. 145-163, 1997.

[27] 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, pp. 5360-5366.

[28] T. H. Cormen, *Introduction to algorithms*: MIT press, 2009.

[29] D. S. Johnson and M. A. Trick, *Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11-13, 1993* vol. 26: American Mathematical Soc., 1996.

[30] K. Xu, "BHOSLIB: Benchmarks with hidden optimum solutions for graph problems (maximum clique, maximum independent set, minimum vertex cover and vertex coloring)â€“hiding exact solutions in random graphs. Web site," *Web site, http://www. nlsde. buaa. edu. en/~ kexu/benchmarks/graphbenchmarks. htm*.

[31] M. Fayaz and S. Arshad, "Clever Steady Strategy Algorithm: A Simple and Efficient Approximation Algorithm for Minimum Vertex Cover Problem," in 2015 13th International Conference on Frontiers of Information Technology (FIT) , pp.277-282.