*Sindh Univ. Res. Jour. (Sci. Ser.) Vol. 48 **(4D)** 17-26 (2016)*

SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

Max Degree Around (MDA) Algorithm: A Smart and Efficient Approximate Algorithm for Vertex Cover and Independent Set Problems

M. FAYAZ, S. ARSHAD, A.S. SHAH*, A. SHAH**

University of Malakand, Khyber Pakhtunkhwa, Pakistan

Received 28rd August 2016 and Revised 17th October 2016

Abstract: The minimum vertex cover (MVC) and maximum independent set (MIS) problems are to be determined in terms of a graph of the small set of vertices, which cover all the edges, and a large set of vertices, no two of which are adjacent. MVC and MIS are notable for its capability of modelling other combinatorial problems and real-world applications. The aim of this paper is twofold: first to investigate failures of the state-of- the-art algorithms for MVC problem on small graphs and second to propose a simple and efficient approximation algorithm for the minimum vertex cover problem. Mostly the state of art approximation algorithms for the MVC problem are based on greedy approaches, or inspired from MIS approaches. Hence, these approaches regularly fail to provide optimal results on specific graph instances. These problems motivated to propose Max Degres Around (MDA) approximation algorithm for the MVC problem. The proposed algorithm is simple and efficient than the other heuristic algorithms for the MVC problem. In this paper, we have introduced small benchmark instances and some state of the art algorithms (MDG, VSA, and MVSA) have been tested along with the proposed algorithm. The proposed algorithm performed well as compared to counterpart algorithms tested on graphs with up to 1000 vertices and 150,000 vertices for Minimum Vertex Cover (MVC) and Maximum Independent Set (MIS).

Keywords: Approximation Algorithm, Approximation Ratio, Benchmark, Minimum Vertex Cover (MVC), NP-Complete, Maximum Independent Set (MIS) Problem, Max Degree Around (MDA)

1. INTRODUCTION

Given an undirected graph G= (V, E) with a vertex set V and an edge set E, least a set for $V' \subseteq V$ such that for each edge(x, y) in G= (V, E) x ϵ V' or y ϵ V' or both, V' is said to be a minimum vertex cover of G= (V, E). The maximum independent set having non-adjacent vertices is known as a maximum set (Fayaz and Arshad, 2015). **(Fig-1b)** shows a graph with 9 vertices; the black vertex set {2,3,4,5} indicates a vertex cover (it is also the minimum vertex cover in this graph) and {1,6,7,8,9} indicates an independent set (it is also the maximum independent set in this graph). Hence, all edges are covered in the graph.

The minimum vertex cover and maximum independent set problems have numerous real life applications, such as; VLSI design, Network Security, ATM Placing, Mobile Tower Placing, Placing Guards in a Museum (Weight and Hartmann 2000, Balaji *et al*. 2010). The minimum vertex cover and maximum independent set problems are allied to some eminent optimization problems such as maximum clique. Clique problem has attracted researchers to way out several problems of areas, such as Coding Theory, Pattern Recognition, Computer Vision, Fault Diagnosis, Protein Structure Alignment, Bioinformatics, Design of Quantum Circuits, Cryptography, Radio Frequency Assignment, Biochemistry and Telecommunications (Baamann, 2003).

There are two ways to tackle the minimum vertex cover and maximum independent set problems; one is the approximation approach, which always provides near optimal solution in a specific amount of time, and the second is complete which gives an exact solution of the problem. The approximation algorithm is best choice for selection, when a near optimal solution is required in reasonable time, (the approximation algorithm does not provide the guarantee of optimal solution). The complete algorithm provides optimal result, but the time it takes to compute is increased exponentially with the size of the problem, complete algorithm work well with small size problems where the exact solution is required with no time constrains (Fayaz and Arshad, 2015).

MVC and MIS are also linked in a straight manner. If the solution of MVC is available for a graph, then MIS of the same can be calculated, such as MIS: = G= (V, E)-MVC and similarly MVC: = G= (V, E)-MIS. Similarly, if MIS algorithm is available then MVC can be calculated and if MVC algorithm is available then MIS can be calculated (Karp, 1972). The approximation ratio is used to quantify the performance of an

++Corresponding author, email: hamaz_khan@yahoo.com

*SZABIST, Islamabad, Pakistan, email: shahsalamss@gmail.com

**International Islamic University Malaysia (IIUM), Kuala Lumpur, Malaysia, email: asadullah@iium.edu.my

approximation algorithm. The approximation ratio can be calculated by (1) and (2) for MVC and MIS respectively.

$$\rho i = A(I) / OPT(I) \geq 1 \quad (1)$$

$$\rho i = A(I) / OPT(I) \leq 1 \quad (2)$$

Where A (I) is the solution of an approximation algorithm and OPT (I) is the optimal solution of a graph. The value of ρi must always be greater or equal to 1 (ρi ≥ 1) for MVC and less or equal to 1 (ρi ≤ 1) for MIS. ρi =1 indicate the optimal solution returned by an approximation algorithm and deviation from 1 indicates a poor solution (Karakotas, 2005).

The minimum vertex cover and maximum independent problems have two versions, i.e. decision and optimization. The decision version is about to find a solution of the desired size of 'k', where 'k' is the vertex cover or independent set of a problem, the answer of decision will be yes or no, but in the optimization version we have to find the best solution from all possible solutions (Balaji, 2010).

In this paper Max Degree Around (MDA) Algorithm is proposed and compared with some small benchmark instances and state of the art algorithms the proposed algorithm has performed well as compared to counterpart algorithms. The remaining structure of the papers is organized as, the background is given in section 2, the proposed algorithm is provided in section 3, section 4 contains experimental results, and finally conclusion and future work is provided in section 5.

2. BACKGROUND

The detailed literature review of some well-known algorithms is provided here, the algorithms have been selected based on efficiency optimality and less computation time. A graph is also provided for each algorithm to inspect the credibility of algorithm towards optimality. In figures 1, 2, 3 and 4, the bold circle indicates that the vertex is in MVC and the empty (nonbold) circle represents that vertex is in MIS.

Clarkson (1983) suggested an approach for Minimum Vertex Cover, named Maximum Degree Greedy (MDG). The MDG is the simplest algorithm for the MVC problem. The MDG is the altered form of set cover approach settled by Chavatal (1979). The MDG includes the maximum degree vertex of a graph to MVC. The total run time complexity for Maximum Degree Greedy is O (E^2) in the worst case. The minimum vertex problem fails to provide optimal result on many instances as in **(Fig-1)**. In **(Fig-1a)** MVC returned by MDG consists of vertices {1, 2, 3, 4, 5} and MIS ={6,7,8,9} and the optimal vertices which cover the given graph are {2, 3, 4, 5} and maximum non adjacent vertices are {6,7,8,9} in **(Fig-1b)**.

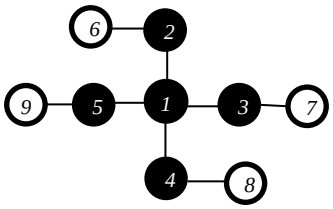


Fig.1. (a) MVC returned by MDG

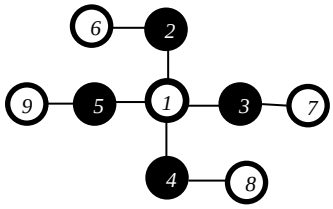


Fig.1. (b) Optimal MVC and MIS

Balaji (2010) proposed Vertex Support Algorithm (VSA). In VSA, new data structure, vertex support is used, which is the sum of all adjacent vertices of a vertex. In this approach, a vertex having maximum vertex support is selected and added to MVC. The VSA simple and is based on greedy approach like MDG, but is more efficient than MDG. It also fails to provide optimal results on some small benchmark graphs like **(Fig-1a)** i.e. returned MVC as MDG.

Imran and Hashim (2013) presented Modified Vertex Support Algorithm (MVSA) and used the same data structure as used in VSA. In MVSA, the minimum support vertex in all adjacent vertices of a minimum support vertex is calculated. The MVSA is not that simple as MDG and VSA, some tweaks have been added to make the algorithm more efficient, that make the algorithm time consuming and more complex. The MVSA takes O (EV^2log v) to compute. The MVSA returned MVC of four vertices {2,3,5,6} and the optimal vertex cover is {1,4,5} and MIS is {1,4}, hence the MVSA fails to return optimal result on the given graph provided in **Fig-2 (a) and (b)**.

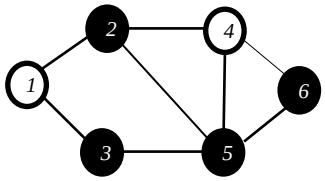


Fig. 2. (a) MVC returned by MVSA

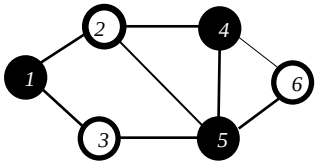


Fig. 2. (b) Optimal MVC and MIS

Gajurel *et al.* (2012) suggested an intelligent approximation algorithm for MVC in which the vertices that are neighbors of the minimum degree vertex have more importance. The NOVCA constructs the vertex cover by repeatedly adding all vertices adjacent to the vertex having minimum degree at each step. In case of a tie, the adjacent vertices of that minimum degree vertex are selected having the maximum sum of degrees of its neighbor degrees. NOVCA also fails to provide optimal result on the graph provided in figure 3 (a) and (b). The run-time complexity of NOVCA is O (V^2 log V). The MVSA returned MVC of vertices {2,3,4,6,8,9,10} and MIS of {1,5,7} and the optimal vertex cover is {1,4,5,7,8,9} and MIS is {2,3,6,10}, hence the NOVACI fails to return optimal solution on the graph.

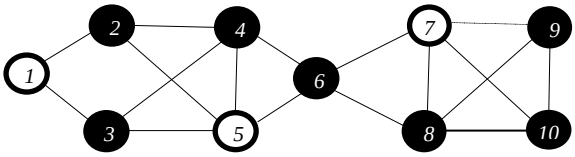


Fig. 3. (a) MVC returned by NOVCA

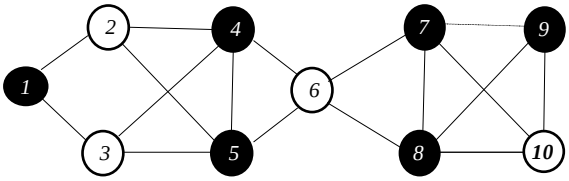


Fig. 3. (b) Optimal MVC and MIS

Gajral and Bielefeld (2012) presented a modified version of NOVCA named NOVCA-II, it also gives importance to adjacent vertices of the minimum degree vertex in a graph, but it breaks the tie by selecting the minimum sum of degrees of its neighbors. The computation time of NOVCA-II is V^2 (V $\log^2$ V). The NOVCA-II returned MVC of four vertices {2,3,4,5} and MIS {1,6} and the optimal vertex cover {1,4,5} and MIS {1,6}, on the graph provided in figure 4 (a) and (b), hence the NOVCA-II fails to return optimal result on the given graph.

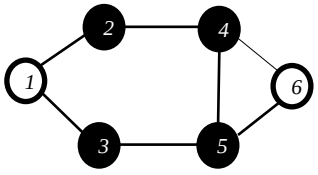


Fig. 4. (a) MVC returned by NOVACA-II

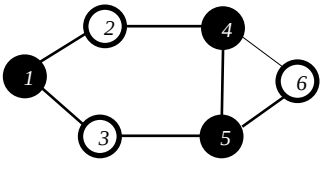


Fig. 4. (b) Optimal MVC and MIS

Beside these, many approximation algorithms have been proposed for solving the minimum vertex cover problem, all these approaches are not suitable for every scenario, such as List Lift, List Right, Divide and Conquer, Ant Colony, (Jovanovic and Tuba (2011), Avis and Imamura (2007), Delbot and Laforest (2008), Asgeirsson *et al.* 2009). These algorithms have some limitations, as some are fast, but poor in optimality and some gives a better approximation ratio, but have high computation time. Some algorithms are designed for the special type of graphs and only provide better result for those graphs, but in general some time we don't know the nature of the graph and it becomes difficult to select algorithm for its solution. Due to the nature of un-exact determination of MVC problem the genetic, bee colony and ant colony algorithms are not suitable to solve it. The divide and conquer algorithm is complete algorithm and provide always optimal result, but the computation time exponentially increases with the size of the problem. The list lift and list right algorithms are fast and provide better optimality but they can be applied to those problems which satisfy some constraints, hence are not applicable for every graph instance. Hence an algorithm which is simple, fast, and do not compromise on optimality is always needed for the MVC problem.The algorithms for MVC, based on greedy strategy are simple and fast, but in many cases, they do not provide optimal results. The algorithms, which are not based on greedy strategy, are not simple and fast, but in many cases, they deliver optimal results. The given graph instances associated with each algorithm help to design an efficient approximation algorithm.

3. PROPOSED ALGORITHM

The proposed Max Degree Around (MDA) up to some level is based on greedy approach, but few intelligent tweaks have been added to make it more efficient. The proposed algorithm is comprised of three steps, in the first step degree calculation of nodes is carried out and checked that a vertex having degree is equal to 1, if it is 1 then the adjacent vertex of that vertex becomes the MVC vertex. In case of more vertices having degree equal to 1, random selection is made and the adjacent vertex of any of them is selected as a candidate for MVC. If the degree of any vertex in the graph is not equal to 1 then the maximum degree vertex is encountered and all its neighbors are located. The algorithm removes edges one by one, and when the degree becomes 1 its neighbor is added to the MVC. After removing all edges between the maximum degree vertex and its neighbor vertices the degree tend to '0' then the maximum degree vertex itself is added to MVC. The pseudo code is provided in subsection 3.2 and flow diagram in **(Fig-5)**, for better understanding of the proposed algorithm.

3.1 Terminologies

Here some terminologies used in the proposed algorithm have been discussed. Let G= (V, E) where V = {v_1, v_2, v_3,……. v_n } and E={e_1, e_2, e_3 ……e_n}, the set of vertices and E is the set of edges. Number of vertices in a graph are denoted by n = |V|, and the number of edges are denoted by m = |E|. Degree of a vertex: d ( ) is used to calculate the degree of a vertex, Neighborhood of a vertex: N (v) = {u ∈ V /u is adjacent to v} where v ∈ V, | | is used to return the number of vertices, Δ and δ symbols are used to represent maximum degree and the minimum degree in G= (V, E) respectively.

In pseudo code, the text of the right side of // is comments for the better elaboration of mathematical notations used in the proposed approach.

3.2 Pseudo Code of MDA

Input: G = (V, E).
Output: C, MIS
Begin:
1. C ←∅;
2. while E ≠ ϕ do {
3. n ← |V|
4. for i ← 1 to n {
i. Calculate d (v_i) }
5. min_d ← δ (G=(V,E))
6. if (min_d =1) {
i. u ←N(md)}
7. Go to 23
8. if (min_d >1){
ii. x ← Δ (G)
ii. p ← |N(x)|
iii. for j← 1 to p {
a). Remove e_j ∈ x and N_j(x)
b). if (d(N_j(x))=1){
c). u ← N(N_j(x))
d). Go to 23}
vi. if (d(x) = 0) {
a). u ← x }
9. Go to 13}}
10. G ← G\u
11. C←C U u
12. } Go to while
13. MIS=V-C
14. Return C
15. Return MIS
End

3.3. Flow Diagram of MDA

The systematic flow chart of the proposed algorithm is provided in **(Fig-5)** for the better understanding of the working mechanism of the proposed Max Degree Around (MDA) algorithm.

4. EXPERIMENTAL WORK AND RESULTS

For the simulation and collaboration, the results of the proposed algorithm and compare with the other algorithms such as MDG, VSA and MVSA, Matlab R2010a version 7.10.0.499 installed on Intel core i5 system having windows 7 is used. We have applied these algorithms along with the proposed algorithm on small benchmark graphs. By employing, an algorithm on small benchmark instances gives the clear picture of the weakness and strengthens of an algorithm. Tackling the weakness and preserve strengthens of these algorithms helped to design a better MDA algorithm for MVC. The proposed algorithm has been tested with large benchmark instances such as on BHOSLIB and DIMACS instances, testing an algorithm on large benchmarks illustrated the overall performance of an algorithm such time complexity, and optimality (Xu, 2012). The proposed algorithm has been inspected on small graphs as given in figure 1, 2, 3 and 4 and on all these graphs it gives optimal results for MVC and MIS.

Graph can be represented in computer memory in many ways, one is adjacency matrix which is an n-by-n matrix, M,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 common graph data structure is simply an edge list – a list (or a set) of edges requires O (m) space and can be represented by a set of pairs. In the proposed approach, the edge list data structure has been used (Cormen, 2009).

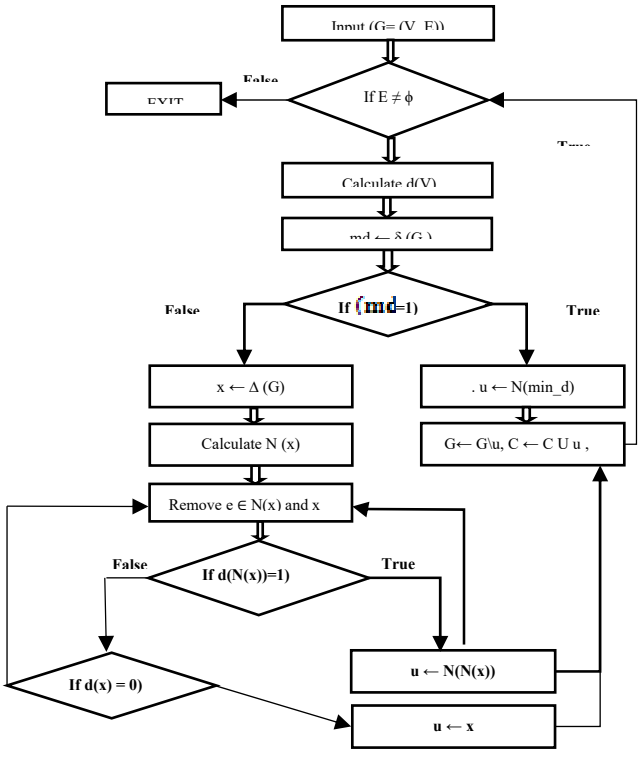


Fig.5. Flow Diagram of the Proposed Algorithm (MDA)

4.1. Results for BHOSLIB and DIMACS benchmark graphs

The practical experiments are necessary for proper evaluation of the algorithm. The performance

comparison of three algorithms is carried out with the proposed algorithm on the BHOSLIB and DIMACS benchmarks for MVC, and presented in the **(Table-1).**

The results are demonstrated in detail that how the proposed algorithm performs on instances of DIMACS and BHOSLIB benchmarks.In **(Table-2)**, the MIS have been calculated by using the number of vertices returned by MDA, MDG, VSA and MVSA for MVC on BHOSLIB and DIMACS benchmark instances. The pproximation ratio is also calculated. In **(Table-1)**, the greater value of an approximation ratio indicates poor performance of algorithm, but in **(Table-1)**, it is opposite, the less approximation ratio indicates the poor result.

4.2. Performance Comparison by the Time

The time comparison of MDA, MDG, VSA and MVSA is presented in the table 3. The MDG and VSA both are difficult by only considering time due to their inherent simplicity (it does not require any complex sub-procedure). In terms of computation time, the proposed MDA algorithm has performed almost same as MDG and VSA, but better than MVSA.

Table-1. Results with DIMACS and BHOSLIB Benchmark Graphs for MVC

Sr. No	Benchmarks instances	V	Opt MVC	MDA MVC	MDG MVC	VSA MVC	MVSA MVC	MDA MVC ρ	MDG MVC ρ	VSA MVC ρ	MVSA MVC ρ
1	graph50_6	50	38	38	38	44	38	1	1	1.158	1
2	graph50_10	50	35	35	35	41	35	1	1	1.171	1
3	graph100_1	100	60	60	60	95	60	1	1	1.583	1
4	graph100_10	100	70	70	70	96	70	1	1	1.371	1
5	graph200_5	200	150	150	150	184	150	1	1	1.227	1
6	graph500_1	500	350	350	350	485	350	1	1	1.386	1
7	graph500_2	500	400	400	400	484	400	1	1	1.210	1
8	graph500_5	500	290	290	290	454	290	1	1	1.565	1
9	phat300-1	300	292	292	293	292	294	1	1.003	1	1.007
10	phat300-2	300	275	276	278	275	279	1.0036	1.010	1	1.015
11	phat300-3	300	264	265	269	264	272	1.0037	1.018	1	1.030
12	phat700-1	700	689	691	693	689	692	1.0028	1.005	1	1.004
13	phat700-2	700	656	657	660	656	660	1.003	1.001	1	1.006
14	phat700-3	700	638	640	642	638	649	1.0031	1.006	1	1.017
15	johnson8-2	28	24	24	24	24	24	1	1	1	1
16	johnson8-4	70	56	56	62	56	56	1	1.107	1	1
17	johnson16-2	120	112	112	112	112	112	1	1	1	1
18	johnson32-2	496	480	480	480	480	480	1	1	1	1
19	sanr200_0.7	200	182	183	184	182	186	1.0054	1.010	1	1.021
20	sanr200_0.9	200	158	162	164	158	163	1.0253	1.037	1	1.031
21	sanr400_0.5	400	387	388	392	387	389	1.0025	1.021	1	1.005
22	sanr400_0.7	400	379	381	384	379	381	1.0052	1.021	1	1.005
23	fbr35-17-2	595	560	563	570	573	564	1.0053	1.021	1.021	1.009
24	fbr_30_15_5	450	420	424	429	429	524	1.009	1.009	1.023	1.008
25	c125	125	91	92	93	91	95	1.003	1.01	1	1.043
26	C250.9	250	206	211	211	206	211	1.0145	1.024	1	1.024
27	C500.9	500	≤443	448	453	443	449	1.02	1.022	1	1.013
28	C2000.9	2000	1922	1925	1944	1923	-	1.002	1.011	1.001	----
29	brock200_1	200	188	189	190	188	191	1.005	1.021	1	1.015
30	brock200_4	200	183	185	192	183	193	1.01	1.027	1	1.054
31	hamming6-2	64	32	32	32	32	32	1	1	1	1
32	hamming6-4	64	60	60	60	60	60	1	1	1	1
33	hamming8-2	256	128	128	128	128	128	1	1	1	1
34	hamming8-4	256	240	240	240	240	240	1	1.033	1	1
35	hamming10-2	1024	512	512	512	512	512	1	1	1	1
36	dsjc-500	500	487	488	491	487	489	1.002	1.008	1	1.004
37	killer4	171	160	160	164	160	160	1	1.025	1	1
38	killer5	776	749	752	764	749	754	1.004	1.020	1	1.006
39	c-fat200-1	200	188	188	188	188	188	1	1	1	1
40	c-fat200-2	200	176	176	176	176	176	1	1	1	1
41	c-fat200-5	200	142	142	142	144	142	1	1	1.014	1
42	c-fat500-1	500	486	486	486	486	486	1	1	1	1
43	c-fat500-2	500	474	474	474	474	474	1	1	1	1
44	c-fat500-5	500	436	436	436	436	436	1	1	1	1
45	c-fat500-10	500	374	374	374	374	374	1	1	1	1
46	MANN_a27.c	378	252	253	261	253	253	1.004	1.003	1.004	1. 004

Table-2. Results with DIMACS and BHOSLIB Benchmark Graphs for MIS

Sr. No.	Benchmark instances	V	Opt MIS	MDA MIS	MDG MIS	VSA MIS	MVSA MIS	MDA MIS ρ	MDG MIS ρ	VSA MIS ρ	MVSA MIS ρ
1	graph50_6	50	12	12	12	6	12	1	1	.5	1
2	graph50_10	50	15	15	15	9	15	1	1	.6	1
3	graph100_1	100	40	40	40	5	40	1	1	.125	1
4	graph100_10	100	30	30	30	4	30	1	1	.1333	1
5	graph200_5	200	50	50	50	16	50	1	1	.32	1
6	graph500_1	500	150	150	150	115	150	1	1	.766	1
7	graph500_2	500	100	100	100	16	100	1	1	.16	1
8	graph500_5	500	210	210	210	46	110	1	1	.219	1
9	phat300-1	300	8	8	7	8	6	1	.875	1	.75
10	phat300-2	300	25	24	22	25	21	.96	.88	1	.84
11	phat300-3	300	36	35	31	36	28	.9722	.86	1	.777
12	phat700-1	700	11	9	7	9	8	.818	.636	1	.72
13	phat700-2	700	44	43	40	44	40	.97	.91	1	.91
14	phat700-3	700	62	60	58	62	51	.96	.93	1	1
15	johnson8-2-4	28	4	4	4	4	4	1	1	1	1
16	johnson8-4-4	70	14	14	8	14	14	1	.51	1	1
17	johnson16-2-4	120	8	8	8	8	8	1	1	1	1
18	johnson32-2-4	496	16	16	16	16	16	1	1	1	1
19	sanr200_0.7	200	18	17	16	18	17	.94	.888	1	.94
20	sanr200_0.9	200	42	38	36	42	37	.90	.86	1	.881
21	sanr400_0.5	400	13	12	8	13	11	.92	.615	1	.84
22	sanr400_0.7	400	21	19	16	21	19	.90	.76	1	.90
23	fbr35-17-2	595	35	33	25	23	30	.94	.71	.65	.85
24	fbr_30_15_5	450	30	26	21	21	26	.86	.7	.7	.86
25	c125	125	34	33	32	34	30	.97	.94	1	.88
26	C250.9	250	44	39	39	44	39	.886	.886	1	.886
27	C500.9	500	57	52	47	57	51	.91	.82	1	1
28	C2000.9	2000	78	75	56	77	-	.96	.71	.98	----
29	brock200_1	200	12	11	10	12	9	.91	.83	1	.75
30	brock200_4	200	17	15	8	17	7	.882	.47	1	.41
31	hamming6-2	64	32	32	32	32	32	1	1	1	1
32	hamming6-4	64	4	4	4	4	4	1	1	1	1
33	hamming8-2	256	128	128	128	128	128	1	1	1	1
34	hamming8-4	256	16	16	16	16	16	1	1	1	1
35	hamming10-2	1024	512	512	512	512	512	1	1	1	1
36	dsjc-500	500	13	12	9	13	11	.92	.69	1	.84
37	killer4	171	11	11	9	11	11	1	.881	1	1
38	killer5	776	27	24	12	27	24	.888	.444	1	.888
39	c-fat200-1	200	12	12	12	12	12	1	1	1	1
40	c-fat200-2	200	14	14	14	14	14	1	1	1	1
41	c-fat200-5	200	68	68	68	66	68	1	1	.97	1
42	c-fat500-1	500	14	14	14	14	14	1	1	1	1
43	c-fat500-2	500	26	26	26	26	26	1	1	1	1
44	c-fat500-5	500	64	64	64	64	64	1	1	1	1
45	c-fat500-10	64	64	64	64	64	6	1	1	1	1
46	MANN_a27.clq.b	378	26	25	17	25	25	.96	.653	.96	.96

Table 3. Time taken by MDA, MDG,VSA and MVSA

Sr. No	Benchmark	V	MDA	MDG	VSA	MVSA
1	graph50_6	50	0.1250	0.1094	0.2031	.4356
2	graph50_10	50	0.1094	0.0938	0.1875	.3321
3	graph100_1	100	0.9063	0.5	1.2031	1.6756
4	graph100_10	100	1.5156	0.6406	1.2656	1.9213
5	graph200_5	200	6.6094	4.9531	9.8594	12.2534
6	graph500_1	500	139.6406	168.25	363.1875	400.239
7	graph500_2	500	549.0469	540.1094	643.5156	703.325
8	graph500_5	500	353.7656	351.3281	738.2969	845.3214
9	Hamming6_2	64	0.125	0.0625	0.2031	.554
10	Hamming6_4	64	0.3594	0.2188	0.3438	.563
11	Hamming8_2	256	1.8906	0.9375	2.375	4.32
12	Hamming8_4	256	24.5156	17.7656	30.8906	40.235
13	Hamming10_2	1024	4.83E+01	53.5625	52.3281	60.32
14	Hamming10_4	1024	3.05E+03	3.03E+03	3.17E+03	3.20E+03
15	jhonson8_4_4	70	0.2813	0.1875	0.2813	.5321
16	jhonson8_2_4	28	0.0313	0.0313	0.0625	.1245
17	jhonson32_2_4	496	731.5938	895.4375	1.16E+03	1.36E+03
18	c 125	125	1.8906	0.3594	0.625	.9354
19	C250	250	5.6563	2.6563	4.25	7.235
20	sanr200-0.7	200	7.6875	3.7188	6.7344	11.231
21	sanr200-0.9	200	3.6719	1.4219	2.4219	3.2155
22	sanr400_0.5	400	112.8281	93.3438	161.1719	171.325
23	sanr400_0.7	400	77.1875	50.8281	94.2188	199.325
24	sanr400_0.5	400	112.8281	93.3438	161.1719	187.213
25	sanr400_0.7	400	77.1875	50.8281	94.2188	111.253
26	gen200_p0.9_44	200	2.5938	1.4063	2.6094	3.215
27	cfat200_1	200	21.0625	11.8438	21.0781	26.324
28	cfat200_2	200	18.2344	10.2969	18.7813	25.213
29	cfat200_5	200	13.3125	5.4844	8.4219	11.215
30	cfat500_1	500	1103	1.06E+03	1.49E+03	1.51E+03
31	cfat500_2	500	107E+03	977.1563	1.38E+03	1.45E+03
32	cfat500_5	500	41.7656	611.8125	786.4063	936.254
33	phat300_1 c	300	51.9375	38.8281	59.8906	70.325
34	phat700_1 c	700	176E+03	2.55E+03	2.40E+03	2.60E+03
35	phat700_2 c	700	195E+03	1.86E+03	2.46E+03	2.60E+03
36	mann_a27 c	378	4.3125	2.0625	3.9844	6.325
37	dsjc500	500	257.8906	256.876	277.993	333.265
38	keller4 c	171	3.625	2.6406	4.2813	10.321
39	keller5 c	776	844.125	894.6719	1227	1345

In **(Table-1)**, the proposed approximation algorithm is compared with some well-known approximation algorithm for MVC and calculated the approximation ratio. The worst and the average approximation ratio of each algorithm is listed in **(Table-4)** and graphically represented in **(Fig-8)**. The worst approximation ratio represents poor optimality and average approximation show the overall performance of approximation algorithm. An algorithm having worst and average approximation ratios near to 1 indicates better performance. It is obvious from the **(Table-4)** that MDA provide better worst and the average of approximation ratio as compared to other counterpart algorithms.

Table. 4. Worst and Average Approximation Ratio

Sr. No	Algorithms	Worst ρi	Average Pi
1	MDA	1.0253	1.002964
2	MDG	1.107	1.012613E0
3	VSA	1.583	1.059435E0
4	MVSA	1.064	1.008

In **(Table-2)**, the proposed approximation algorithm has been compared with some well-known approximation algorithm for MIS. The worst and average approximation ratios of MDA, MDG ,VSA and MVCA for the maximum independent set is given in **(Table-5)**, opposit to MVSA in MIS the highest value means the worst and average approximation ratios values means better performance of the algorithm. The worst and the average approximation ratios are represented in **(Fig-10)**. It is obvious from **table 5** that the performance of MDA is better as compared to counterpart algorithms.

4.3. Graphical Representation of Results

The **(Fig-6)** illustrates the vertices cover by four algorithms diagrammatically for better representation of the performance of MDA against other counterpart algorithms with respect to the number of vertices cover.

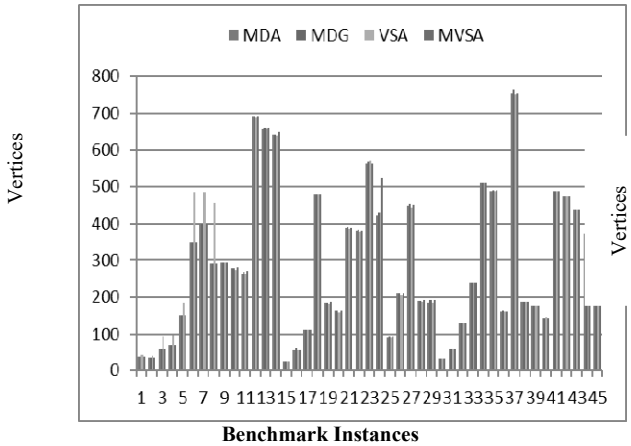


Fig. 6. Number of vertices cover by MDA, MDG, VSA and MVSA

The **(Fig-7)** represents the approximation ratio of each algorithm.

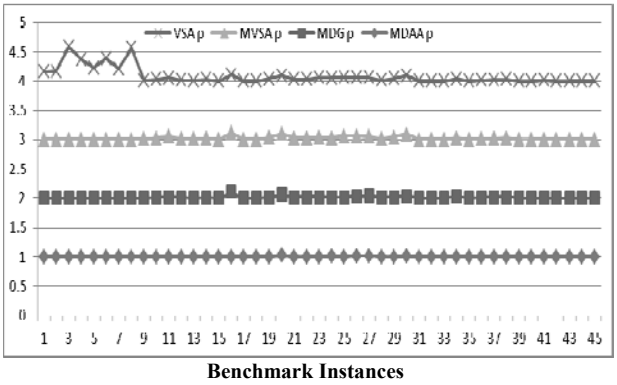


Fig-7. Approximation ration of VSA, MVSA, MDG and Proposed MDA

The worst and average approximation of each algorithm are digramatically represented in **(Fig-8)**.

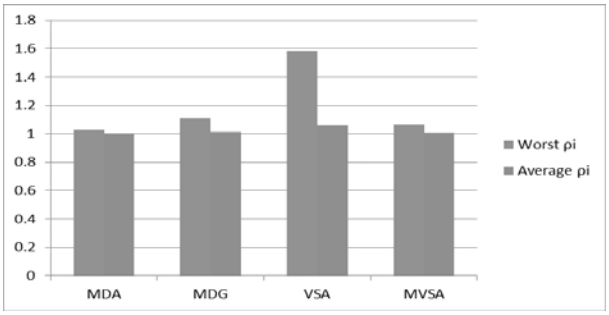


Fig-8. Worst and Average Approximation Ratio of, MDA, MDG, VSA and MVSA

In **(Fig-9)**, the number of vertices in MIS for each algorithm is represented, oppose to MVS here, the high the length of a bar indicates better result. It is clear from the graph that MDA performed well as compared to MDG, VSA and MVSA.

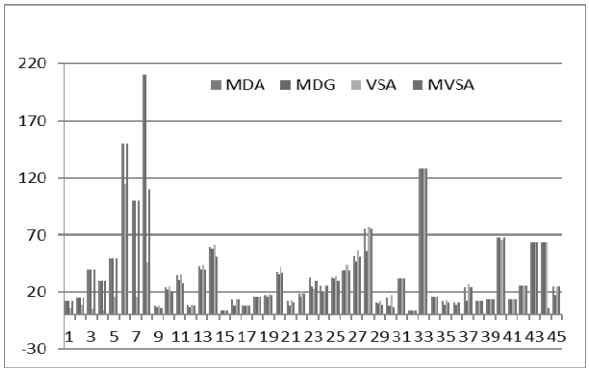


Fig-9. Maximum Independent Sets of MDA, MDG, VSA and MVSA on Benchmark Instances

The worst and the average approximation ratio of each algorithm for MIS is represented in **(Fig-10)**.

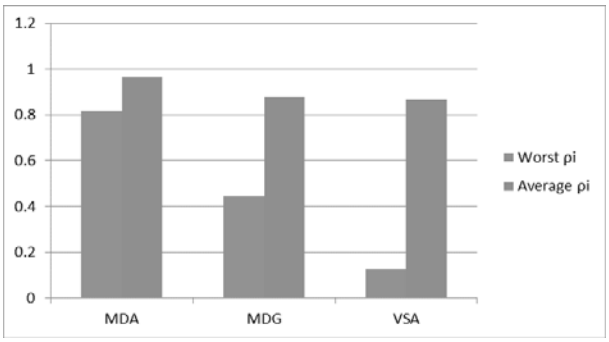


Fig-10. Worst and Average Approximation Ratio of MDA, MDG and VSA for MIS

The time taken by each algorithm is represented in **(Fig-11)**.

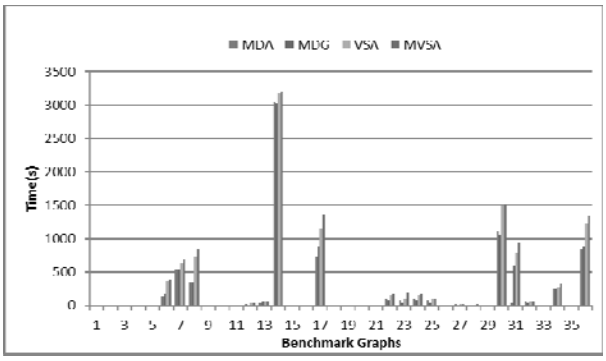


Fig11. Time taken by MDG, VSA, MVSA and MDA

5. CONCLUSION

In this paper, a novel algorithm for finding a minimum vertex cover in a graph is presented. The algorithm deals with NP-problem. The better approximation ratio for different kinds of graphs has great importance in the proposed work. The algorithm provides better average and worst approximation ratio as compared to the other algorithms. The proposed algorithm is tested on small graphs and BHOSLIB and DIMACS benchmark to test its efficiency in terms of optimality and computation time. The current version of the proposed algorithm is for un-weighted MVC, in the future, we will add some tweaks to make it suitable for weighted MVC. Normally, we find the approximation ratio to measure the performance, but the approximation ratio is suitable when the optimal solution of a problem is known. If the optimal solution is not available in advance and we want to measure the performance of approximation algorithm, then it becomes a challenge. In the future, we will provide a strategy to measure the performance of an approximation algorithm in case of non-availability of optimal solution.

REFERENCES:

Avis, D., T. Imamura, (2007). A list heuristic for vertex cover. Operations research letters, 35(2), 201-204.

Asgeirsson, E. I., C. Stein, (2009). Divide-and-Conquer Approximation Algorithm for Vertex Cover. SIAM Journal on Discrete Mathematics, 23(3), 1261-1280.

Baamann, K. (2003). The Maximum Clique Problem-On Finding an Upper Bound with Application to Protein Structure Alignment. Doctoral dissertation, Georgia Institute of Technology.

Balaji, S., V. Swaminathan, and K. Kannan, (2010). Optimization of Un-Weighted Minimum Vertex Cover. World Academy of Science, Engineering and Technology, 4(7), 941-946.

Chvatal, V., (1979). A Greedy Heuristic for the Set-Covering Problem. Mathematics of operations research, 4(3), 233-235.

Clarkson, K. L., (1983). A Modification of the Greedy Algorithm for Vertex Cover. Information Processing Letters, 16(1), 23-25.

Cormen, T. H., (2009). Introduction to Algorithms, MIT press.

Delbot, F., C. Laforest, (2008). A Better List Heuristic for Vertex Cover. Information Processing Letters, 107(3-4), 125-127.

Fayaz, M., S. Arshad, (2015). 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), 277-282.

Gajurel, S., R. Bielefeld, (2012). A Simple NOVCA: Near Optimal Vertex Cover Algorithm. Procedia Computer Science, 9, 747-753.

Gajurel, S., R. Bielefeld, (2012). A Fast Near Optimal Vertex Cover Algorithm (NOVCA). International Journal of Experimental Algorithms (IJEA), 3(1), 9-18.

Imran, K., K. Hasham, (2013). Modified Vertex Support Algorithm: A New Approach for Approximation of Minimum Vertex Cover. Research Journal of Computer and Information Technology Sciences,1(16),1-6.

Jovanovic, R., M. Tuba, (2011). An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem. Applied Soft Computing, 11(8), 5360-5366.

Karakostas, G. (2005). A Better Approximation Ratio for the Vertex Cover Problem. In: Proceedings of the 32nd International Colloquium on Automata, Languages and Programming, ICALP, 1043–1050.

Karp, R. M. (1972). Reducibility Among Combinatorial Problems, Springer.

Weigt, M., A. K. Hartmann, (2000). Number of Guards Needed by a Museum: A Phase Transition in Vertex Covering of Random Graphs. Physical review letters, 84: 6118.

Xu, K., (2012). Vertex Cover Benchmark Instances (DIMACS & BHOSLIB). IJEA (international journal of Experimental algorithms), 3(1), 1-18.