A Smart Approximation Algorithm for Minimum Vertex Cover Problem based on Min-to-Min (MtM) Strategy

Jawad Haider[1], Muhammad Fayaz[2]
Department of Computer Science
University of Central Asia
Naryn 722918, Kyrgyzstan

***Abstract*—In this paper, we have proposed an algorithm based on min-to-min approach. In the proposed algorithm first the degree of each vertex of the graph is calculated. Next the vertex with minimum degree is selected, after which all the neighbors of the minimum degree are located. In the neighbors of the minimum degree vertex, again the vertex with the minimum degree is found and put into the set minimum vertex cover and deleted from the graph. Again, the degree of each vertex of the updated graph is calculated and again the same process is repeated until the graph becomes empty. In case of tie, all the neighbors of the minimum degree vertices are computed and then the minimum degree vertex in all of them is added to minimum vertex degree set. The same process is repeated until the graph becomes empty. The proposed algorithm is a very simple, efficient, and easy to understand and implement. The proposed min-to-min algorithm is evaluated on small as well as on large benchmark instances and the results indicate that the performance of the min-to-min algorithm is far better as compared to the other state-of-art algorithms in term of accuracy and computation complexity. We have also used the proposed method to solve the maximum independent set problem.**

Keywords*—Minimum vertex cover; approximation algorithms; maximum independent set; benchmark instances; graph theory

I. Introduction

A graph in field of computer science is the set of vertices, and collection of edges each of which connects a pair of vertices [1]. Edges are the links between the vertices. Mathematically, a graph is represented as G (V, E), where V denotes the set of vertices and E the collection of edges[2] .In graph theory a vertex cover is defined as the subset Vc $\subseteq$ V, such that the vertices in the subset Vc covers all the edges E in the graph G(V, E). Minimum vertex cover(MVC) is the minimum number of elements in the subset Vc which covers all the edges in the graph [3]. The minimum vertex cover problem is well known in graph theory because of its real-life applications in diverse fields. For example, MVC has application in civil and electrical engineering, map labeling, sensor networks as well as in very-large scale integration (VLSI) design, bioinformatics and biochemistry, protein sequencing and gene regulatory network [4-7]. A real-life example to better explain the application of MVC is the positions of guards in a museum. Where each edge of the graph represents the corridors of the museum and each vertex denotes the position of the guards. To station minimum guards while still covering all the corridors of the museum is minimum vertex cover problem [8].

Minimum vertex cover problem comes in two versions-optimization version and decision version. The decision version is a Boolean type problem. Where the question is to find, if there exists a solution of desired size k? k is the minimum number of vertices that should be used. And the result is either true or false (Yes or No). The optimization version is all about finding the optimal solution[5]. Our concern in this paper will be with second version i.e optimization version of MVC.

In 1979 S.Cook put two conditions for NP problems to be called as NP-complete. 1) It should be NP-hard and 2) it should be reducible to any NP-complete problem in deterministic polynomial time [9]. Cook put the Boolean Satisfiability (SAT) problem as the basic problem in the set of NP-complete problems. As clique can be converted into SAT problem in polynomial time, so clique belongs to SAT problems. Moreover, maximum independent set (MIS) problem can be converted to clique problem as well, similarly minimum vertex cover can be converted into MIS problem in polynomial time, hence the above problems are interchangeable and are considered as NP-complete problems [10]. To conclude, minimum vertex cover (MVC) problem is NP-complete problem. MIS, MC and MVC are all complimentary concepts and NP-complete problems.

Maximum independent set is also a great problem, where the optimizing task is to find a set containing maximum number of such vertices of the graph where no pair of the vertices share an edge or are adjacent to each other [11]. Its application can be found in map labelling problems where the names of adjacent cities should be placed nearest to the city without any overlap [7]. MIS is also used in high level synthesis and physical design automation, while computing the MIS in a graph can be used to determine the maximum number of processors required for parallel execution and also used in channel routing problems of physical design automation, for instance k-layer routing for Printed Circuit Boards and Multi-Chip Modules [12]. Moreover, MIS have application in information retrieval, classification theory, scheduling, economics, computer vision and experimental design [13].

There are two types of algorithms to solve any NP-complete problems. These are the approximation algorithms and exact algorithms [14,15]. The exact algorithms will always provide a solution that is optimal, but the computation time increases exponentially with the size of the problem. Therefore, the exact algorithms are best option for small size problems where an optimal solution is needed without any time constraints. Brute force algorithm, branch and bound algorithm and divide and conquer algorithms come under this type, where all the possible solutions are evaluated and the optimal one is selected [16, 17]. On the other hand, the approximation algorithms come as the best option after the exact algorithms for solving NP-complete problems. Greedy algorithms, simple heuristic algorithms and memetic algorithms are example of approximation algorithms. As stated earlier, the execution time increases exponentially with the size for NP-complete problems using exact algorithms, even for an ordinary size NP-complete problem it takes thousands or billions of execution time years to compute the solution using the currently available computational power. While the approximation algorithms provide an approximate result in polynomial runtime. Therefore, the approximation algorithms are preferably the better option for researcher to come up with a solution to NP-complete problems [18,19]. The proposed algorithm -min-to-min is also an approximation algorithm to solve MVC problem. We used Approximation ratio - the tool to evaluate the performance of any approximation algorithms.

This paper is extended version of our paper published [1]. In this paper we have proposed a new algorithm using the min-to-min (MtM) approach. The purpose of the proposed algorithm is to enhance the optimality and decrease the computational time. The developed approach is very simple and is based on simple heuristic method. Due to simplicity and intelligent selection of vertices in vertex cover set it save the time and improve the performance in term of optimality of the proposed algorithm.

This paper is structured as such: first the related work is discussed in detail in Section II and then the proposed algorithm is elaborated in Section III. While Section IV presents the implementation, alone with our experimental results, and discussion. Finally, Section V presents the conclusion of the paper in detail. The abbreviations with their corresponding descriptions are presented in detail in Table I.

TABLE I. ABBREVIATIONS WITH DESCRIPTIONS

Notations	Descriptions
MVC	Minimum Vertex Cover
MIS	Maximum Independent Set
MDG	Maximum Degree Greedy
VSA	Vertex Support Algorithm
MVSA	Modified Vertex Support Algorithm
MtM	Min-to-Min
AE	Absolute Error
V	Vertices
E	Edges

II. RELATED WORK

Several approximation algorithms were proposed by various authors to deal with the problem of minimum vertex cover. Based on their performance in terms of their run time complexity, optimality, and performance on small benchmarks, a brief literature review of these popular algorithms proposed for MVC are discussed here.

The first algorithm is maximum degree greedy algorithm (MDG). The approach adopted in this algorithm is greedy. It introduces few changes in the previously existing greedy heuristic algorithm for set-cover problem by Chavatal in 1979 [20]. It uses the idea of subtracting the weight of covered vertex from all its neighbor vertices into the greedy algorithm for finding the vertex cover. Therefore, it adds all the vertices having the maximum degree to MVC. Its run time complexity for the worst case is $O(E^2)$. This algorithm fails even on small benchmarks as shown in Fig. 1(b).

On the other hand, different approaches were also used to create more efficient algorithm. Vertex Support Algorithm (VSA) is one of them.

VSA is also an approximation algorithm. The basic version of this algorithm is that it uses an adjacency matrix with binary variables 1,0; to indicate the existence of edge between two vertices with 1, and absence with 0. Its output shows if any vertex is included in vertex cover or not included. A new data structure – support of a vertex- is implemented in the algorithm. It is defined as the sum of the degree of the adjacent vertices to a given vertex of the graph. It calculates the degree and support of all the vertices, then select those vertices which have maximum support and put into vertex cover. If there are vertices with same support, then it selects the maximum degree vertex, and includes that vertex to vertex cover set [5]. The VSA has the run time complexity of $O(EV^2)$ in the worst-case scenario.

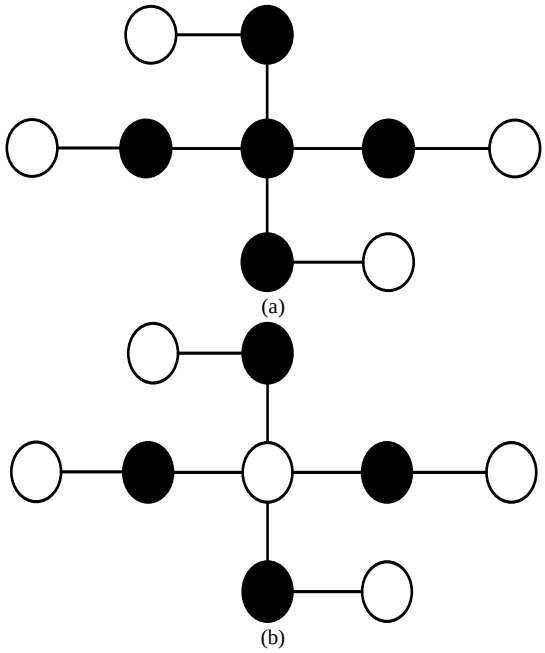


Fig. 1. (a) Optimal MVC, and (b) MVC through MDG, MVSA and VSA.

Furthermore, there is an improved and modified version of VSA named as modified vertex support algorithm (MVSA). It was proposed by Imran *et al*. in 2013[21]. The modification introduced by Imran *et al*. is in the criteria of selecting the vertex for considering it as candidate for minimum vertex cover, which, as seen above in MDG is selecting the maximum degree vertex, while in VSA the vertex which has maximum support value was selected. MVSA works in three stages. First is the analysis step, where the values of degrees and support of the vertices is calculated. Then comes the filtering process where the values are categorized as 'min_support'- vertices which have the minimum value for support and 'adj_nodes'- all the adjacent vertices to the vertices having minimum support values. And then in the last step selection is done by choosing the vertex with minimum support value, as candidate vertex for MVC. Again, the run time complexity for this algorithm is $O(EV^2 \log V)$.

Another modified version of vertex support algorithm proposed by Ahmad *et al.* [22] is known as advanced vertex support algorithm (AVSA). Ahmad *et al* introduced a tiny change in MVSA. Keeping the same data structure of support of vertex, while instead of selecting the maximum or minimum support of vertex as seen above in VSA and MVSA respectively, an alternate way was adopted by just finding all the adjacent vertex to those vertices which have minimum support. Then from all the found adjacent vertices, again selecting the one with minimum support. The computational complexity for AVSA is same as MVSA i.e. $O(EV^2 \log V)$.

Other than the above algorithms, Gujrat *et al* [23] have also contributed to the problem of minimum vertex cover by proposing the near optimal vertex cover algorithm (NOVCA). It works by successively appending the adjacent vertices to the minimum degree vertex in each step. It deals with a tie situation by choosing the neighbors of a vertex having the maximum degree instead of using neighbor of minimum degree vertex. This algorithm also has the polynomial run time complexity of $O(EV^2 \log V)$. Using NOVCA on small benchmark instances yielded failure.

In 2012, Gujral *et al* [24] further improved their algorithm and came up with a modified version of NOVCA i.e NOVCA-II. NOVCA-II is simply the incorporation of idea that a vertex cover should be built by adding the vertex in decreasing order of their degree and in the situation of tie, it uses the converse of NOVCA i.e. it chooses the vertices which have the minimum sum of degrees of their neighbor vertex, instead of selecting the neighbors of vertex having maximum degree as in NOVCA. By using few graph instances, it became evident that both algorithms of Gujral *et al* also fail on small benchmarks.

More investigation has been done and Li *et al* in 2011[25] came up with a new idea of max-share of degree heuristic algorithm (Max-I). This algorithm works based on the concept that, only those vertices should be added to minimum vertex cover, which helps to reduce the degree of its neighboring vertices as much as possible, or ultimately if zero. It follows random selection in cases of tie. Max-I has the worst run time complexity of $O(n^3)$.

However, in 2014, Imran *et al* [26] again came up with his second method for MVC called Degree Contribution Algorithm (DCA). A new data structure -Degree contribution- was introduce. Degree contribution is used for graph processing parameters. It helps in taking the complete graph to determine the values for each vertex. Degree contribution of a vertex is the total number of degrees of the vertex and sum of all the vertices with same degree in the whole graph. The degree contribution for each vertex determines either it is efficient to choose that vertex for MVC or not. So, DCA works by calculating the degree of every vertex along with its degree contribution value**.** Then only those vertices are added to minimum vertex cover whose degree contribution value is higher than others. After adding a vertex with higher degree contribution value, all its neighbor edges are removed and the same steps are repeated for the remaining vertices, until no edge was left in the graph. The execution time of DCA is $O(VE)$. A memetic algorithm was also proposed by Jovanovic *et al* in 2011 [27].

Based on heuristic nature of real ants searching for food, ant colony algorithm was proposed for minimum weight vertex cover problem (MWVC). As memetic framework of ant colony, bee colony and bat colony were mostly used for optimization problems. However, the ant colony algorithm was initially applied to the traveling salesman problems (TSP), where a fully connected graph with weighted edges were considered. While in case of MVC it is not necessary for the graph to be fully connected and un-weighted. So, Jovanovic *et al*. used arbitrary edges to make the given graph a connected graph and gave weight to each vertex to make it complacent for the algorithm. Hence, converting the graph from MVC to TSP and applying the ant colony algorithm was the main aim. For un-weighted graphs, this algorithm is much harder to apply. Lastly an unusual attempt in proposing a unique algorithm for MVC was undertaken by Halldórsson *et al* in [28]. Based on the greedy strategy the greed is good (GGA) algorithm was proposed. This algorithm tackled the maximum independent set (MIS) problem. The "Network Bench Model" was used in this algorithm. The degree for every vertex is found then only the minimum degree vertex is included in MIS. Once, MIS is found then MVC can be easily fond by taking complement of MIS from the set of all vertices.

Li et al. [29] proposed a new local search technique to solve the minimum vertex cover problem called NuMVC. NuMVC comprised of three main stages. In the first stage the introduction of four rules is carried out. In the second stage a technique is introduced called configuration checking in order to reduce cycling in local search. In the third phase a method is introduced to reduce the searching time in order to improve the optimality and reduce the computation time. In [30] two new extensions of the conventional vertex cover problem are introduced. The detail presentation of the two new methods is carried out in detail.

In the literature review section different algorithms are discussed in detail from different aspects with the aim to find the weakness and strengths of the existing algorithms for minimum vertex cover problem. These algorithms have weakness in one way or another way, some are fast but fail to provide optimal results on many cases, some algorithms

provide good optimality results but are very slow. The proposed method is introduced to tackle all these issues.

III. Proposed Algorithm

The algorithm put forth here, has a different and unique approach than any algorithms found in the literature review. The min-to-min (MtM) consists of three basic steps of working through an undirected and unweighted graph with any number of vertices. The initial step in the algorithm is to find the degree of every vertices in the provided graph, then comes the next step of finding the minimum degree vertex and getting its adjacent vertices. If there are more vertices with same minimum degree than we will choose the first vertex with minimum degree. After getting the minimum degree vertex and all of its adjacent vertices. The last step is to again find the minimum degree vertex from the adjacent vertices of already found minimum degree vertex, and this vertex is considered as a candidate for MVC. Once the candidate vertex is selected from the adjacent vertices, all its edges are deleted, and the vertex is appended to MVC. This process is repeated till no edge is left in the provided graph.

A. Data Structure

In our work, we implemented the edge list data structure for representing graphs in computer memory leveraging the properties of saving computation time for sparse graphs using list indexing to access any edge and also fully exploiting the advantages of saving space as well instead of using adjacency list for the graph representation. In case of the space usage the performance of edge list is $O(x + y)$ for the representation of the graph having *x* edges and *y* vertices. And in case of the run time, edge list performs well in counting the number of vertices or edges and looping an iteration over the edges or vertices [31]. The iterations over the edge list in our case using the method find () has a runs time of $O(1)$. While the method min(array) iterates through the required array in run time of $O(y)$, y being the total vertices. Furthermore, removing any edges associated with a vertex has the complexity of run time $O(x)$, which reduces exponentially as by each iteration our graph gets smaller with few vertices and associated edges. On the other hand, adjacency matrix- which is mostly used as data structure for graph representation- has both performance drawbacks of run time as well as space storage problems for our purpose. For instance, the $O(x^2)$ space usage is the worse than the $O(x + y)$ space usage by edge list, moreover most real-world graphs are sparse which makes it more disadvantageous to implement adjacency matrix as depicted in Fig. 2(a) and (b) the adjacency matrix and edge list representation respectively of the graph from Fig. 1(a).

	1	2	3	4	5	6
1	0	1	1	0	0	0
2	1	0	0	1	1	0
3	1	0	0	1	1	0
4	0	1	1	0	1	1
5	0	1	1	1	0	1
6	0	0	0	1	1	0

(a)

1	2
1	3
2	4
2	3
4	5
4	6
5	6

(b)

Fig. 2. Ardency Matrix and Edge List.

B. Terminologies

Following are some terminologies that we have used in the proposed pseudo code as illustrated in Fig. 3 deg (): used for degree calculation, | |: used for absolute values, min(): for minimum degree calculation in the graph, adj (): to calculate the neighbors/adjacent vertices of a node/vertex and C is used for cover.

Pseudo code for Clever Steady Strategy Algorithm

Start:
Input: = G (V, E)
Output: = C
FOR i← 1 to n **{**
$d_i \leftarrow deg(V_i)$
}
While G ≠ θ **{**
$M_i \leftarrow min(d_i)$
IF $|M_i| = 1$ **{**
$N_i \leftarrow adj\ (M_i)$
}
IF $|M_i| = 1$ **{**
$N_i \leftarrow adj\ (M_i)$
$MN_i \leftarrow adj\ (M_i)$
$C \leftarrow adj\ (M_i)$
}
}
Display C
Stop

Fig. 3. Pseudo Code of the Min-to-Min Algorithm.

C. Flow Chart of the Algorithm

The flow chart of the proposed algorithm is provided in Fig. 4. In the flow diagram we have used the same terminologies as given in the pseudo code of the proposed MtM algorithm.

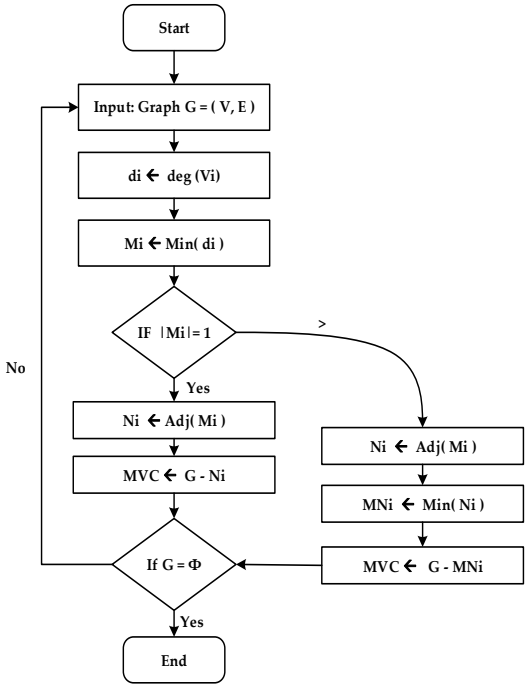


Fig. 4. Flow Chart of the Proposed Algorithm.

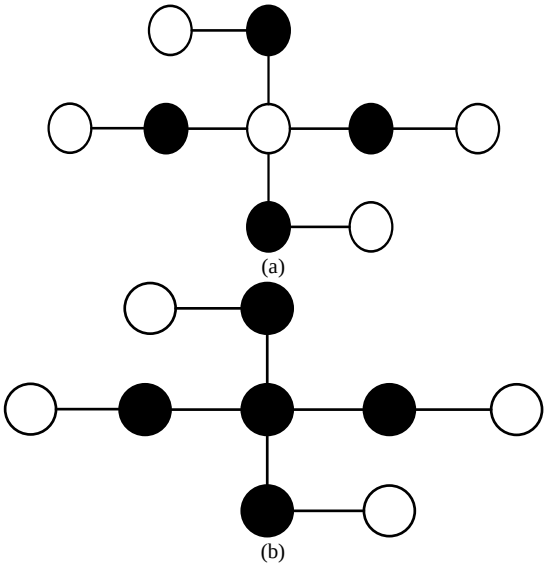
IV. Implementation, Empirical Results and Discussion

A. Implementation Setup

Here, the section deals with the empirical results gained through implementation of MtM on various benchmark instances of up-to-date popular libraries. By cross evaluating the results of MtM to different well-known proposed algorithms, as discussed in literature review also, the tables were created. And by the run time complexity and optimality the result of MtM were compare with VSA, MDG and MVSA in the tabular form. The coding of the MtM algorithm has been done in MATLAB R2014a version 7.10.0.499 on an Intel core i5 system having windows 7 operating system.

B. Results on Small Instances

Implementing the proposed algorithm on small benchmark revealed an interesting result and optimal performance of MtM as compare to performance of other algorithms discussed in the literature review. As shown in Fig. 5(a) the MVC by MtM for the given graph is 4 vertices which is optimal and the approximation ratio ρi = 1. Which is also optimal. While in Fig. 5(b) the same graph is covered by 5 vertices according to MDG and ρi = 1.250. while the given graph in Fig. 5(c) is covered using just 2 vertices by MtM, which is optimal solution and ρi = 1, while it can be seen in Fig. 5(d) the same graph is covered by VSA using 3 vertices with ρi of 1.5. Moreover, most astonishing performance of MtM appears in Fig. 5(e) which is covered by just 1 vertex- which is the optimal solution. But the same graph shown in Fig. 5(f) is covered using 4 vertices by MVSA with ρi = 4. As the solution by MtM for the small benchmark instances is optimal and it outperformed all other existing algorithms, hence the MtM is much efficient as well as fast in relation to the already proposed algorithms in literature.



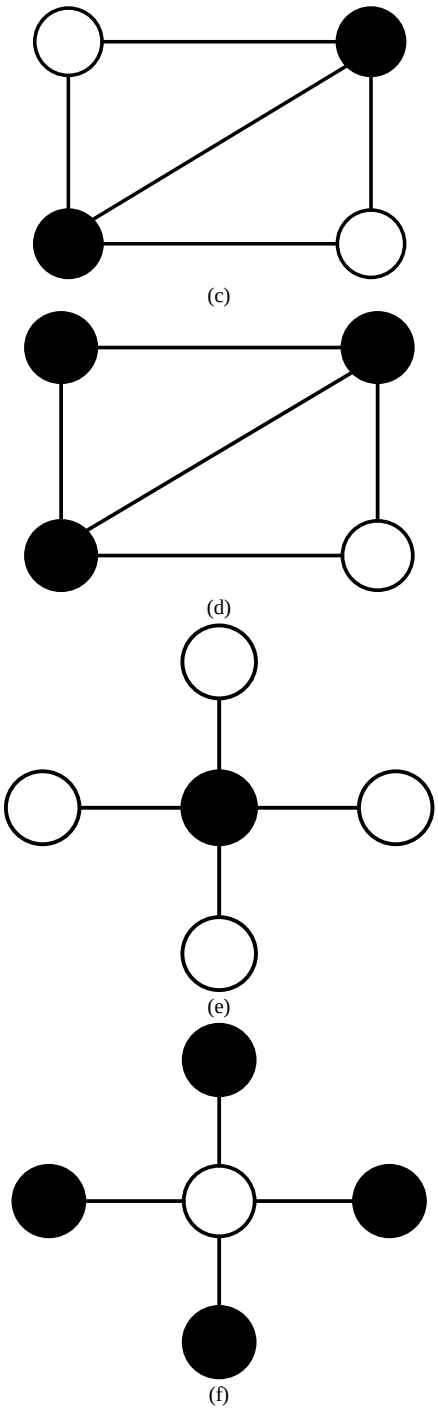


Fig. 5. (a) MtM ρi = 1, (b) MDG ρi = 1.25, (c) MtM ρi = 1, (d) VSA ρi = 1.5, (e) MtM ρi = 1, (f) MVSA ρi = 4.

C. Perfomance of MtM on Large Benchmark Instances

Similarly, the performance by MtM was evaluated on large benchmarks instance along with the comparison of its performance with other well-known existing algorithms. The used libraries for the large benchmark instances were DIMACS and BHOSHLIB. In Table II the performance of MtM is evaluated against VSA, MDG and MVSA based on the total vertices covered by the proposed algorithm along with the counterpart algorithms. The approximation ratio of

each algorithm for each benchmark instance as well as the absolute error was evaluated in Table III. In Table VI the worst-case approximation ratios and average cases approximation ratios have been calculated for MtM, VSA, MDG and MVSA for MVC. The approximation ratios of MtM for both cases are much smaller in comparison with the other state of art algorithms which indicates that the working of MtM algorithm is much efficient than the counterpart algorithms in literature. Similarly, the approximation ratios for the worst and average case have been calculated for MtM, MDG, VSA, and MVSA for maximum independent set problem (MIS) in Table VII. The approximation ratios also indicate that the results from the proposed method are efficient as compared to the other algorithms for MIS problem.

TABLE II. PERFORMANCE OF MTM, MDG, VSA AND MVSA BENCHMARK INSTANCES FOR MINIMUM VERTEX COVER (MVC)

S. NO	Benchmarks	V	C*	MtM	MDG	VSA	MVSA
				|MVC|	|MVC|	|MVC|	|MVC|
1	graph50_6	50	38	38	38	44	38
2	graph50_10	50	35	35	35	41	35
3	graph100_1	100	60	60	60	95	60
4	graph100_10	100	70	70	70	96	70
5	graph200_5	200	150	150	150	184	150
6	graph500_1	500	350	350	350	485	350
7	graph500_2	500	400	400	400	484	400
8	graph500_5	500	290	290	290	454	290
9	phat300-1	300	292	293	293	292	294
10	phat300-2	300	275	275	278	275	279
11	phat300-3	300	264	266	269	264	272
12	phat700-1	700	689	692	693	689	692
13	phat700-2	700	656	658	660	656	660
14	phat700-3	700	638	640	642	638	649
15	johnson8-2-4	28	24	24	24	24	24
16	johnson8-4-4	70	56	56	62	56	56
17	johnson16-2-4	120	112	112	112	112	112
18	johnson32-2-4	496	480	480	480	480	480
19	sanr200_0.7	200	182	183	184	182	186
20	sanr200_0.9	200	158	164	164	158	163
21	sanr400_0.5	400	387	388	392	387	389
22	sanr400_0.7	400	379	382	384	379	381
23	fbr35-17-2	595	560	565	570	573	424
24	fbr_30_15_5	450	420	426	429	429	565
25	c125	125	91	94	93	91	95
25	C250.9	250	206	211	211	206	211
27	C500.9	500	443	448	453	443	449
28	C2000.9	2000	1922	1927	1944	1923	1937
29	brock200_1	200	188	190	190	188	191
30	brock200_4	200	183	185	192	183	193
31	gen200_p0.9_44	200	156	164	165	156	166
32	hamming6-2	64	32	32	32	32	32
33	hamming6-4	64	60	60	60	60	60
34	hamming8-2	256	128	128	128	128	128
35	hamming8-4	256	240	240	240	240	240
36	hamming10-2	1024	512	512	512	512	512
37	dsjc-500	500	487	489	491	487	489
38	killer4	171	160	160	164	160	160
39	killer5	776	749	754	764	749	754
40	c-fat200-1	200	188	188	188	188	188
41	c-fat200-2	200	176	176	176	176	176
42	c-fat200-5	200	142	142	142	144	142
43	c-fat500-1	500	486	486	486	486	486
44	c-fat500-2	500	474	474	474	474	474
45	c-fat500-5	500	436	436	436	436	436
46	c-fat500-10	500	374	374	374	374	374
47	MANN_a27.clq.b	378	252	253	261	253	253

TABLE III. PERFORMANCE AND COMPARISON OF MTM, AND MDG, VSA, MVSA USING APPROXIMATION RATIO AND ABSOLUTE ERROR FOR MVC

S. NO	Approximation Ratio				Absolute Error			
	MtM	MDG	VSA	MVSA	MtM	MDG	VSA	MVSA
1	1.0000	1.0000	1.1579	0.8636	0	0	6	0
2	1.0000	1.0000	1.1714	0.8537	0	0	6	0
3	1.0000	1.0000	1.5833	0.6316	0	0	35	0
4	1.0000	1.0000	1.3714	0.7292	0	0	26	0
5	1.0000	1.0000	1.2267	0.8152	0	0	34	0
6	1.0000	1.0000	1.3857	0.7216	0	0	135	0
7	1.0000	1.0000	1.2100	0.8264	0	0	84	0
8	1.0000	1.0000	1.5655	0.6388	0	0	164	0
9	1.0034	1.0000	0.9966	1.0068	1	1	0	2
10	1.0000	1.0109	0.9892	1.0145	0	3	0	4
11	1.0076	1.0113	0.9814	1.0303	2	5	0	8
12	1.0044	1.0014	0.9942	1.0044	3	4	0	3
13	1.0030	1.0030	0.9939	1.0061	2	4	0	4
14	1.0031	1.0031	0.9938	1.0172	2	4	0	11
15	1.0000	1.0000	1.0000	1.0000	0	0	0	0
16	1.0000	1.1071	0.9032	1.0000	0	6	0	0
17	1.0000	1.0000	1.0000	1.0000	0	0	0	0
18	1.0000	1.0000	1.0000	1.0000	0	0	0	0
19	1.0055	1.0055	0.9891	1.0220	1	2	0	4
20	1.0380	1.0000	0.9634	1.0316	6	6	0	5
21	1.0026	1.0103	0.9872	1.0052	1	5	0	2
22	1.0079	1.0052	0.9870	1.0053	3	5	0	2
23	1.0089	1.0088	1.0053	0.7400	5	10	13	136
24	1.0143	1.0070	1.0000	1.3170	6	9	9	145
25	1.0330	0.9894	0.9785	1.0440	3	2	0	4
25	1.0243	1.0000	0.9763	1.0243	5	5	0	5
27	1.0113	1.0112	0.9779	1.0135	5	10	0	6
28	1.0026	1.0088	0.9892	1.0073	5	22	1	15
29	1.0106	1.0000	0.9895	1.0160	2	2	0	3
30	1.0109	1.0378	0.9531	1.0546	2	9	0	10
31	1.0513	1.0061	0.9455	1.0641	8	9	0	10
32	1.0000	1.0000	1.0000	1.0000	0	0	0	0
33	1.0000	1.0000	1.0000	1.0000	0	0	0	0
34	1.0000	1.0000	1.0000	1.0000	0	0	0	0
35	1.0000	1.0000	1.0000	1.0000	0	0	0	0
36	1.0000	1.0000	1.0000	1.0000	0	0	0	0
37	1.0041	1.0041	0.9919	1.0041	2	4	0	2
38	1.0000	1.0250	0.9756	1.0000	0	4	0	0
39	1.0067	1.0133	0.9804	1.0067	5	15	0	5
40	1.0000	1.0000	1.0000	1.0000	0	0	0	0
41	1.0000	1.0000	1.0000	1.0000	0	0	0	0
42	1.0000	1.0000	1.0141	0.9861	0	0	2	0
43	1.0000	1.0000	1.0000	1.0000	0	0	0	0
44	1.0000	1.0000	1.0000	1.0000	0	0	0	0
45	1.0000	1.0000	1.0000	1.0000	0	0	0	0
46	1.0000	1.0000	1.0000	1.0000	0	0	0	0
47	1.0040	1.0316	0.9693	1.0000	1	9	1	1

The Absolute error is calculated for each algorithm on each benchmark instance, in Table III. Similarly, the MtM is also compared with other algorithms in terms of getting the Maximum independent set in Table IV. And the approximation ratio along with Absolute error for MIS of each algorithm is presented in Table V.

Mathematically Absolute Error is defined as, the magnitude of the difference between the measured/ obtained or approximated value of a quantity $x_{approx.}$ and the actual or optimal value $x_{optimal}$.

$$\Delta X = \left| x_{optimal} - x_{approx} \right| \quad (1)$$

In our case, absolute error ΔX served as the best indicator of the performance of the algorithm. Like the value ΔX shows how close the result of the algorithm is to the actual solution. The smaller the value of absolute error ΔX, the better the solution can be considered. Large value of ΔX indicate worst performance and a poor solution. Table III shows the approximation ratio, along with Absolute error values, which are calculated for the measurement of performance with respect to the optimal output of each algorithm.

TABLE IV. PERFORMANCE OF MTM, MDG, VSA AND MVSA BENCHMARK INSTANCES FOR MAXIMUM INDEPENDENT SET (MIS)

S. NO	Benchmarks	V	C*	MtM	MDG	VSA	MVSA
				|MIS|	|MIS|	|MIS|	|MIS|
1	graph50_6	50	12	12	12	6	12
2	graph50_10	50	15	15	15	9	15
3	graph100_1	100	40	40	40	5	40
4	graph100_10	100	30	30	30	4	30
5	graph200_5	200	50	50	50	16	50
6	graph500_1	500	150	150	150	15	150
7	graph500_2	500	100	100	100	16	100
8	graph500_5	500	210	210	210	46	210
9	phat300-1	300	8	7	7	8	6
10	phat300-2	300	25	25	22	25	21
11	phat300-3	300	36	34	31	36	28
12	phat700-1	700	11	8	7	11	8
13	phat700-2	700	44	42	40	44	40
14	phat700-3	700	62	60	58	62	51
15	johnson8-2-4	28	4	4	4	4	4
16	johnson8-4-4	70	14	14	8	14	14
17	johnson16-2-4	120	8	8	8	8	8
18	johnson32-2-4	496	16	16	16	16	16
19	sanr200_0.7	200	18	17	16	18	14
20	sanr200_0.9	200	42	36	36	42	37
21	sanr400_0.5	400	13	12	8	13	11
22	sanr400_0.7	400	21	18	16	21	19
23	fbr35-17-2	595	35	30	25	22	171
24	fbr_30_15_5	450	30	24	21	21	-115
25	c125	125	34	31	32	34	30
25	C250.9	250	44	39	39	44	39
27	C500.9	500	57	52	47	57	51
28	C2000.9	2000	78	73	56	77	63
29	brock200_1	200	12	10	10	12	9
30	brock200_4	200	17	15	8	17	7
31	gen200_p0.9_44	200	44	36	35	44	34
32	hamming6-2	64	32	32	32	32	32
33	hamming6-4	64	4	4	4	4	4
34	hamming8-2	256	128	128	128	128	128
35	hamming8-4	256	16	16	16	16	16
36	hamming10-2	1024	512	512	512	512	512
37	dsjc-500	500	13	11	9	13	11
38	killer4	171	11	11	7	11	11
39	killer5	776	27	22	12	27	22
40	c-fat200-1	200	12	12	12	12	12
41	c-fat200-2	200	24	24	24	24	24
42	c-fat200-5	200	58	58	58	56	58
43	c-fat500-1	500	14	14	14	14	14
44	c-fat500-2	500	26	26	26	26	26
45	c-fat500-5	500	64	64	64	64	64
46	c-fat500-10	500	126	126	126	126	126
47	MANN_a27.clq.b	378	126	125	117	125	125

Approximation ratio is the ratio of number of vertices in MCV found by an algorithm to the optimal solution of the MVC. Mathematically, approximation ratio is defined- as in Eq. (2).

$$\rho_i = \frac{A_i}{OPT_i} \quad (2)$$

In the above equation, A denotes the approximate result and *i* represents the number of instances, OPT*i* represents the optimal result.

For MVC $\rho i = 1$ is the optimal solution while for the existing algorithms $\rho i \geq 1$ always. A ratio very close to 1 means the result found is better and almost near to optimal solution while the more the ratio deviates from 1, the solution gets poor and worst. On the other hand, for MIS $\rho i = 1$ also means the result provided by the algorithm is optimal but all existing algorithms will always have $\rho i \leq 1$, as for MVC for MIS if the deviation from 1 is very small then the solution is considered as the near to optimal, but a deviation of 0.1 means the solution is not better one while high values of the deviation indicates the solution is the worst. Thus, for MIS approximation ratio ≤1 and for MVC approximation ratio ≥ 1.

TABLE V. PERFORMANCE OF MTM, MDG, VSA AND MVSA BASED ON APPROXIMATION RATIO AND ABSOLUTE ERROR FOR MIS

S. NO	Approximation Ratio				Absolute Error			
	MtM	MDG	VSA	MVSA	MtM	MDG	VSA	MVSA
1	1.0000	1.0000	0.5000	1.0000	0	0	6	0
2	1.0000	1.0000	0.6000	1.0000	0	0	6	0
3	1.0000	1.0000	0.1250	1.0000	0	0	35	0
4	1.0000	1.0000	0.1333	1.0000	0	0	26	0
5	1.0000	1.0000	0.3200	1.0000	0	0	34	0
6	1.0000	1.0000	0.1000	1.0000	0	0	135	0
7	1.0000	1.0000	0.1600	1.0000	0	0	84	0
8	1.0000	1.0000	0.2190	1.0000	0	0	164	0
9	0.8750	0.8750	1.0000	0.7500	1	1	0	2
10	1.0000	0.8800	1.0000	0.8400	0	3	0	4
11	0.9444	0.8611	1.0000	0.7778	2	5	0	8
12	0.7273	0.6364	1.0000	0.7273	3	4	0	3
13	0.9545	0.9091	1.0000	0.9091	2	4	0	4
14	0.9677	0.9355	1.0000	0.8226	2	4	0	11
15	1.0000	1.0000	1.0000	1.0000	0	0	0	0
16	1.0000	0.5714	1.0000	1.0000	0	6	0	0
17	1.0000	1.0000	1.0000	1.0000	0	0	0	0
18	1.0000	1.0000	1.0000	1.0000	0	0	0	0
19	0.9444	0.8889	1.0000	0.7778	1	2	0	4
20	0.8571	0.8571	1.0000	0.8810	6	6	0	5
21	0.9231	0.6154	1.0000	0.8462	1	5	0	2
22	0.8571	0.7619	1.0000	0.9048	3	5	0	2
23	0.8571	0.7143	0.6286	4.8857	5	10	13	136
24	0.8000	0.7000	0.7000	-3.8333	6	9	9	145
25	0.9118	0.9412	1.0000	0.8824	3	2	0	4
25	0.8864	0.8864	1.0000	0.8864	5	5	0	5
27	0.9123	0.8246	1.0000	0.8947	5	10	0	6
28	0.9359	0.7179	0.9872	0.8077	5	22	1	15
29	0.8333	0.8333	1.0000	0.7500	2	2	0	3
30	0.8824	0.4706	1.0000	0.4118	2	9	0	10
31	0.8182	0.7955	1.0000	0.7727	8	9	0	10
32	1.0000	1.0000	1.0000	1.0000	0	0	0	0
33	1.0000	1.0000	1.0000	1.0000	0	0	0	0
34	1.0000	1.0000	1.0000	1.0000	0	0	0	0
35	1.0000	1.0000	1.0000	1.0000	0	0	0	0
36	1.0000	1.0000	1.0000	1.0000	0	0	0	0
37	0.8462	0.6923	1.0000	0.8462	2	4	0	2
38	1.0000	0.6364	1.0000	1.0000	0	4	0	0
39	0.8148	0.4444	1.0000	0.8148	5	15	0	5
40	1.0000	1.0000	1.0000	1.0000	0	0	0	0
41	1.0000	1.0000	1.0000	1.0000	0	0	0	0
42	1.0000	1.0000	0.9655	1.0000	0	0	2	0
43	1.0000	1.0000	1.0000	1.0000	0	0	0	0
44	1.0000	1.0000	1.0000	1.0000	0	0	0	0
45	1.0000	1.0000	1.0000	1.0000	0	0	0	0
46	1.0000	1.0000	1.0000	1.0000	0	0	0	0
47	0.9921	0.9286	0.9921	0.9921	1	9	1	1

TABLE VI. APPROXIMATION RATIOS FOR WORST AND AVERAGE CASE FOR MTM, MDG, MVSA, AND VSA FOR MVC

Algorithms	Worst case for MIS ρ_i	Average for MIS ρ_i
MtM	0.7272	0.9476
MDG	0.4444	0.8804
VSA	0.1000	0.8602
MVSA	-3.8333	0.9010

TABLE VII. APPROXIMATION RATIOS FOR WORST AND AVERAGE CASE FOR MDG, MVSA AND VSA FOR MIS

Algorithms	Worst case for MVC ρ_i	Average for MCV ρ_i
MtM	1.0512	1.0054
MDG	1.1071	1.0065
VSA	1.5833	1.0467
MVSA	1.3170	0.9681

V. Conclusion and Future Work

This paper proposes a new algorithm – Min to Min (MtM) to tackle the NP- hard optimization problems of MVC as well as MIS problems of graph theory. The algorithm was well tested first on small benchmark instances as well as then it was investigated on benchmarks instance of libraries like DIMACS and BHOSLIB. The obtained experimental results were cross compared with other extant algorithms of literature. After the obtained results it was proved that MtM outperformed algorithms like MDG, VSA and MVSA on the parameters of runtime performance as well as approximation ratio and absolute error. The results are well demonstrated using tables and visual graph diagrams. Hence, the proposed algorithm is an efficient way to solve real world problems of MVC and MIS in different areas of application. It is simple and fast to implement, and it gives near to optimal solutions on the grounds of run time, storage and parameter of approximation ratio, and the absolute error. While it outperforms other existing algorithms in worst and average cases.

In the future we would like to add some more tweaks to the proposed algorithm to improve it in term of optimality and apply on other NP-complete problems as such as graph coloring, maximum clique etc.

Acknowledgment

The authors would like to acknowledge Professor Neal Gallagher who motivated and encouraged us for research at the University of Central Asia. Further, the authors would like to thank Sumaino Shakarbekova for proofreading the paper.

References

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

[2] D. B. West, Introduction to graph theory. Prentice hall Upper Saddle River, NJ, 1996.

[3] V. Kann, "On the approximability of NP-complete optimization problems," Royal Institute of Technology Stockholm, 1992.

[4] F. N. Abu-Khzam, M. A. Langston, P. Shanbhag, and C. T. J. A. Symons, "Scalable parallel algorithms for FPT problems," vol. 45, no. 3, pp. 269-284, 2006.

[5] S. Balaji, V. Swaminathan, K. J. W. A. o. S. Kannan, Engineering, and Technology, "Optimization of unweighted minimum vertex cover," vol. 43, pp. 716-729, 2010.

[6] S. Richter, M. Helmert, and C. Gretton, "A stochastic local search approach to vertex cover," in Annual Conference on Artificial Intelligence, 2007, pp. 412-426: Springer.

[7] B. Verweij and K. Aardal, "An optimisation algorithm for maximum independent set with applications in map labelling," in European Symposium on Algorithms, 1999, pp. 426-437: Springer.

[8] M. Weigt and A. K. J. P. r. l. Hartmann, "Number of guards needed by a museum: A phase transition in vertex covering of random graphs," vol. 84, no. 26, p. 6118, 2000.

[9] S. A. Cook, "The complexity of theorem-proving procedures," in Proceedings of the third annual ACM symposium on Theory of computing, 1971, pp. 151-158.

[10] R. M. Karp, "Reducibility among combinatorial problems," in Complexity of computer computations: Springer, 1972, pp. 85-103.

[11] M. Xiao and H. J. D. A. M. Nagamochi, "An exact algorithm for maximum independent set in degree-5 graphs," vol. 199, pp. 137-155, 2016.

[12] D. Kagaris and S. Tragoudas, "Maximum independent sets on transitive graphs and their applications in testing and CAD," in International Conference on Computer Aided Design: Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design, 1997, vol. 9, no. 13, pp. 736-740.

[13] T. A. Feo, M. G. Resende, and S. H. J. O. R. Smith, "A greedy randomized adaptive search procedure for maximum independent set," vol. 42, no. 5, pp. 860-878, 1994.

[14] M. Garey and D. Johnson, "Computers and Intractability: A guide to the Theory of NP-Completeness, WH Freedman, San Francisco," 1979.

[15] M. R. Garey, D. S. J. C. Johnson, and Intractability, "A Guide to the Theory of NP-Completeness," pp. 37-79, 1990.

[16] E. Balas and J. J. A. Xue, "Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring," vol. 15, no. 5, pp. 397-412, 1996.

[17] R. Carraghan and P. M. J. O. R. L. Pardalos, "An exact algorithm for the maximum clique problem," vol. 9, no. 6, pp. 375-382, 1990.

[18] L. A. Hall and D. S. Hochbaum, "Approximation lgorithms for NP-hard problems, chapter approximation algorithms for scheduling," ed: PWS Publishing Company, Boston, MA, 1996.

[19] G. Karakostas, "A better approximation ratio for the vertex cover problem," in International Colloquium on Automata, Languages, and Programming, 2005, pp. 1043-1050: Springer.

[20] V. J. M. o. o. r. Chvatal, "A greedy heuristic for the set-

[21] covering problem," vol. 4, no. 3, pp. 233-235, 1979.

[22] K. Imran, K. J. R. J. o. C. Hasham, and I. T. S.

[23] "Modified Vertex Support Algorithm: A New approach for approximation of Minimum vertex cover," vol. 2320, p. 6527, 2013.

[24] I. Ahmad, M. J. I. J. o. A. S. Khan, and Technology, "AVSA, modified vertex support algorithm for approximation of MVC," vol. 67, pp. 71-78, 2014.

[25] S. Gajurel and R. J. P. C. S. Bielefeld, "A Simple NOVCA: Near Optimal Vertex Cover Algorithm," vol. 9, pp. 747-753, 2012.

[26] S. Gajurel and R. J. I. J. o. E. A. Bielefeld, "A fast near optimal vertex cover algorithm (novca)," vol. 3, pp. 9-18, 2012.

[27] S. Li, J. Wang, J. Chen, and Z. J. J. Wang, "An Algorithm for Minimum Vertex Cover Based on Max-I Share Degree," vol. 6, no. 8, pp. 1781-1788, 2011.

[28] I. K. a. H. Khan, "Degree Contribution Algorithm for Approximation of MVC," International Journal of Hybrid Information Technology, vol. 7, no. 5, pp. 183-190, 14 October 2014 2014.

[29] R. Jovanovic and M. J. A. S. C. Tuba, "An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem," vol. 11, no. 8, pp. 5360-5366, 2011.

[30] M. M. H. a. J. Radhakrishnan, "Greed is Good: Approximating Independent Sets in Sparse and Bounded-Degree Graphs," Algorithmica, vol. 18, pp. 145-163, 1997. Springer-Verlag New York Inc.

[31] Li, R., Hu, S., Cai, S., Gao, J., Wang, Y., & Yin, M. (2020). NuMWVC: A novel local search for minimum weighted vertex cover problem. Journal of the Operational Research Society, 71(9), 1498-1509.

[32] Akrida, E. C., Mertzios, G. B., Spirakis, P. G., & Zamaraev, V. (2020). Temporal vertex cover with a sliding time window. Journal of Computer and System Sciences, 107, 108-123.

[33] R. T. Michael T. Goodrich, Michael H. Goldwasser, "Graph Algorithms," in Data Structures & Algorithms in Python: John Wiley & Sons, Inc., 2013, pp. 628-633.