O P E R A T I O N S R E S E A R C H A N D D E C I S I O N S
No. 4 2015
DOI: 10.5277/ord150401

Imran KHAN[1]
Naveed RIAZ[2]

A NEW AND FAST APPROXIMATION ALGORITHM FOR VERTEX COVER USING A MAXIMUM INDEPENDENT SET (VCUMI)

The importance of non-deterministic polynomial (NP) problems in real world scenarios has compelled researchers to consider simple ways of finding approximate solutions to these problems in polynomial time. Minimum vertex cover is an NP complete problem, where the objective is to cover all the edges in a graph with the minimal number of vertices possible. The maximal independent set and maximal clique problems also belong to the same class. An important property that we have analyzed while considering various approaches to find approximate solutions to the minimum vertex cover problem (MVC) is that solving MVC directly can result in a bigger error ratio. We propose a new approximation algorithm for the minimum vertex cover problem called vertex cover using a maximum independent set (VCUMI). This algorithm works by removing the nodes of a maximum independent set until the graph is an approximate solution of MVC. Based on empirical results, it can be stated that VCUMI outperforms all competing algorithms presented in the literature. Based on all the benchmarks used, VCUMI achieved the worst case error ratio of 1.033, while VSA, MDG and NOVAC-1 gave the worst error ratios of 1.583, 1.107 and 1.04, respectively.

Keywords: *non-deterministic polynomial, vertex cover, independent set, benchmark, error ratio*

Shortcuts

NP – non-deterministic polynomial
MVC – minimum vertex cover
MIS – maximum independent set
VSA – vertex support algorithm
DC – degree contribution

[1]Department of Computer Science, University of Swat, Pakistan, e-mail: imran.khan@gssi.infn.it
[2]College of Computer Science and IT, University of Dammam, Saudi Arabia,
e-mail**:** nrmohammed@ud.edu.sa

MWVC – minimal weighted vertex cover
MWIS – maximal weighted independent set
VCUMI – vertex cover using maximum independent set

1. Introduction

A graph is composed of nodes and edges. Vertices or nodes are connected to each other by edges. Many real life problems can be modeled using graphs and after modeling, various techniques can be applied to optimize a specific objective function according to the field of application. MVC can be applied to wireless communication and complex networks [8], civil and electrical engineering, as well as various areas of biochemistry, e.g. bioinformatics, molecular biology [3].

One problem associated with the graph theory is that many problems in this field are intractable, i.e. cannot be solved in polynomial time and the majority of such problems belong to a class called NP-complete. As it is widely believed that exact solutions to NP-complete problems cannot be found in polynomial time, various alternative approaches have been considered by researchers to solve these problems. These techniques are either based on complex heuristics or approximation of the exact solution. However, there is often no guarantee that heuristic solutions produce a quality solution in a reasonable amount of time. On the other hand, approximation techniques always produce an approximate solution in polynomial time. It is pertinent to mention that the quality of a solution can be described by the approximation ratio. In the case of a minimization problem, the approximation ratio is defined as the ratio of the cost of the approximate solution to the cost of the optimal solution,

$$\rho_i = \frac{A(i)}{OPT(i)} \geq 1$$

where i is an instance of the problem, A is the cost of the approximate solution and OPT is the cost of the optimal solution. Note that ρ_i is the approximation ratio for a particular instance i of a problem and

$$\rho_n = \max \rho_i \text{ over all } n,$$

i.e. ρ_n is the maximum value over all the ρ_i (a measure describing the worst case scenario). When $\rho_i = 1$, then the approximate solution is actually optimal. However, in many cases optimality cannot be achieved, since some such problems are intractable. According to Gray and Johnson, the problem is intractable if it is so hard that no poly-

nomial algorithm can possibly solve it [14]. The value of ρ_i describes the quality of a solution, the more it deviates from 1, the poorer is the solution.

Vertex cover is a graph related problem where the objective is to select a set of vertices of a graph that covers all the edges of the graph (i.e. all the edges have at least one end at one of the vertices in this set). Minimal vertex cover is achieved when the total number of vertices in the vertex cover set is minimized. Let a graph G be defined as $G(V, E)$, where V represents the set of vertices and E represents the set of edges. Then a vertex cover V_c is defined as $V_c \subseteq V$ such that V_c covers all the edges in the graph. Figure 1 is a simple graph and the optimal MVC for the graph is depicted in Fig. 2 (the shaded circle).

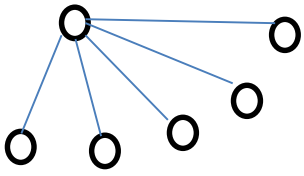

Fig. 1. $G(V, E)$

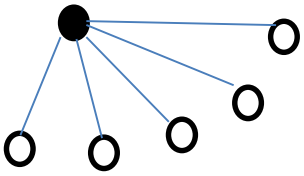

Fig. 2. MVC

Karp showed that finding the minimal vertex cover of a graph is an NP-complete problem [18]. Thus, it is obvious that we cannot get an optimal solution to MVC in polynomial time, unless it is true that P = NP. Due to the existence of a wide range of real life problems that can be formulated as MVC, various approximation and heuristic techniques have been developed and deployed by researchers.

Demaine in [11] argued that vertex cover remains NP-complete even in the case of cubic graphs and according to [13] this holds even in the case of planar graphs of degree at most 3. Current heuristics for solving MVC only consider the features of each vertex in isolation, in order to decide whether a vertex belongs to the solution set or not [20].

Dinur and Safra argued that MVC cannot be guaranteed to give an approximation factor below 1.36, unless P = NP [12]. Numerous techniques and approximation algorithms have been presented in the literature, such as the greedy approach, list left, list right, vertex support algorithm etc. but all of these have limitations in one way or another. Some are reliable but complex. Some are simple but underperform when we take computational complexity into account. Some are simple and fast, but not reliable, i.e. the approximate solutions are poor. Some have a worst case approximation factor of 2 (i.e. are 2-approximations), while some are Δ-approximations, where Δ is variable. Generally, 2-approximation algorithms are considered acceptable.

As vertex cover is NP-complete, so all other NP-complete problems can be reduced to it, due to the property of NP-completeness. The problems of deriving maximum cliques and maximum independent sets are also NP-complete problems that are very closely related to minimum vertex cover. Let $G = (V, E)$ be a graph, where V

denotes the set of vertices in graph G and E denotes the set of edges in G. An independent set of vertices in G is composed of vertices which are not directly connected to each other, formally defined as: "An independent set of a graph $G = (V, E)$ is a subset $V' \subseteq V$ of vertices such that each edge in E is incident on at most one vertex in V', the independent-set problem is to find a maximum-size independent set in G" [6]. The problem of finding a maximum independent set in a graph is NP-complete. Vertex cover is closely related to independent sets, because if V_C is a set of MVC nodes for the graph (V, E), then the remaining nodes are an independent set, i.e. $V - V_C$ forms an independent set. So by finding a way to solve the MVC problem, one can solve the MIS problem and vice versa.

In this paper, we present a new algorithm for approximating a minimum vertex cover by making use of a maximum independent set. The algorithm proposed here is named vertex cover using a maximum independent set (VCUMI). This approach has been tested using some well-known benchmark graphs for demonstrating the efficacy of algorithms. We have also compared the results of VCUMI with MDG and NOVAC, two well-known algorithms that use an approach similar to ours. These results show a considerable improvement over other competing approaches, both in terms of average and worst case solutions, when applied using the same benchmarks.

This paper is organized as follows. The Literature Review section presents previous research work performed by other authors in this field, followed by the Proposed Algorithm section, where we discuss the new algorithm and describe it using a pseudo code which also enables a derivation of the runtime complexity. In the Empirical Results section, we compare the empirical results obtained using the proposed algorithm with several existing approaches. This section also provides a brief overview of the runtime complexity involved in solving the benchmark problems. The final section gives some conclusions from the work presented here.

2. Literature review

Karp proved that the minimal vertex cover problem is NP-complete [19]. This means that we cannot find an optimal solution in polynomial time for the minimal vertex cover problem, or any other problems in this class, if it is true that $P \neq NP$, as is strongly believed in the computer science community. The vertex cover problem has received considerable attention from researchers and scientists in the last decade, due to its potential for an enormous range of applications.

Approximation algorithms are methods used for obtaining solutions to problems that belong to the NP-complete class. This approach is based on the idea of obtaining a near optimal solution in polynomial time, rather than waiting for an unreasonable amount of time to obtain an optimal solution. Numerous approximation algorithms

have already been proposed for solving the minimum vertex cover problem. The author of [4] proposed one of the earliest and simplest approximation algorithms for solving the MVC problem. According to this algorithm, the decision regarding whether to include a vertex in a candidate solution for MVC is based on the minimal available information [4]. At each step, the algorithm selects a single vertex in the input graph via a randomization procedure. Then this random node is added to the MVC and all its adjacent edges are deleted from the input graph. This process is terminated when no edge remains in the graph. This algorithm was later improved by Clarkson [5], who noticed that the nodes included in the vertex cover tend to have relatively high degrees. He opted for an inductive algorithm, where at each step the algorithm selects the node with the maximum degree and then erases the entire set of edges incident to that vertex. This process continues till no edge remains. This algorithm is known as the maximum degree greedy (MDG) and the complexity of this algorithm is $O(E^2)$, where E is the total number of edges in the graph. According to [3], its maximum error is Δ, where Δ is the maximum degree of a vertex in the graph. Experimental analysis carried out in [10] showed that MDG gives a maximum error of 33% on ErdosRenyi graphs, 9% on trees, 44% on Bhoslib, 32% on regular graphs and 70% on average worst case graphs. Worst case graphs are those that are designed to trap an algorithm.

Halldorsson, and Radhakrishnan also took a greedy approach when they proposed an algorithm for constructing a maximum independent set (MIS) called greedy independent cover (GIC) [16]. Their algorithm takes advantage of the simple relation between the maximum independent set and minimum vertex cover problems. GIC works by searching for the node of minimum degree, adding it to the MIS and then erasing all of that node's neighbors. This process continues in an iterative fashion until no node remains unprocessed. The authors of [16] proved that GIC finds the optimal solution for trees and therefore also for paths. The authors of [10] showed that, on average, GIC performs better than other algorithms in terms of quality, but its performance is worst for the average worst case graphs [12]. They also noted, after applying GIC to several categories of graphs, that the error percentage never exceeds 25.

The list left heuristic approach was proposed for solving MVC with a maximum approximation ratio of $(\Delta/2)^{1/2} + 3/2$ and minimum ratio of $p\ \Delta/2$, where Δ is the maximum degree in the graph [7]. They further prove that $(\Delta/2)^{1/2}$ is the smallest possible approximation ratio for any algorithm based on an ordered list that is applied to find a solution of the minimum vertex cover problem. Their algorithm makes use of list heuristics and bases each decision on the information available for a particular instance. An annealing algorithm for deriving a minimum vertex cover was proposed by Xu and Ma [24]. They proposed the adoption of a new acceptance criterion for including a node in the minimum vertex cover. They call each vertex a neuron and the total number of neurons is equal to the total number of vertices in a graph. Delbot and Laforest presented another list heuristic for vertex cover named List Right [9]. This

approach is based on the technically similar List Left that was presented earlier [7] but this algorithm outperforms the older version in terms of approximation ratio and results. This approach gives a worst case approximation ratio of Δ where Δ is the maximum degree in the graph. List Right works by scanning the list of vertices from right to left and deciding whether to include the currently scanned vertex in the MVC.

Asgeirsson and Stein [1] presented a divide and conquer approach but their approach does not seem to be feasible for practical purposes as the criterion for graph division is vague and introduces unrequired complexity. The idea is to divide the graph into subgraphs and then solve the problem for each subgraph.

A different approach is proposed in [2] that selects vertices for the MVC on the basis of a new criterion, named support of a vertex. The support of a vertex is defined to be the sum of the degrees of all its adjacent vertices, i.e.

$$\text{Support}(V) = S(V) = \sum u \in N(v) d(u)$$

A recently proposed algorithm, named NOVAC-I [15], is based on the very important concept of analyzing the relationship between vertices. The observed that vertices which are connected to the vertex of minimum degree very often appear in the MVC. The modified vertex support algorithm (MVSA) proposed by [2], is an adaptation of VSA using a more subtle criterion for including a vertex in the minimum vertex cover. The original method was solely dependent on a greedy methodology, in which they selected a vertex on the basis of maximum support. Rather than selecting a vertex just on the basis of maximum support, MVSA selects a vertex based on an analysis of the support values of all the vertices adjacent to the candidate node. All the vertices adjacent to the vertex with the minimum support value are analyzed first and the vertex from this set with the maximum support is selected. Although this seems to be a small modification, the experimental results presented in [17] show that MVSA can provide better results in comparison to the original VSA [2]. A new approach to vertex support [18], advocates that appropriate analysis of the support value can have a great impact on the final decision.

A hybrid approach to approximating the minimum vertex cover [22] based on a combination of a steady state genetic algorithm with a greedy heuristic. First the genetic algorithm produces a set of nodes, which is then reduced by greedy heuristics. They tested their approach against ant colony optimization (ACO) and showed that their approach not only works faster than ACO, but is also efficient in producing final output.

3. Proposed algorithm

After our analysis of various greedy approaches and the properties of vertices, we note that trying to solve the vertex cover problem directly can increase the approxima-

tion ratio. Hence, we have taken an indirect approach and designed an algorithm for deriving a maximum independent set. From such a solution, we can easily get a vertex cover and/or maximal clique. The steps of our algorithm are

We first calculate the degree of each vertex in the input graph and then identify those nodes in the graph that have a degree of one. These nodes are potential candidates for MIS based on the following criteria:

- If there is a single node of degree one, then we add it to the 'MIS' set, add its neighbor to the 'MVC' set and delete these two nodes from the graph. Otherwise, if there is more than one node of degree one, then we select one at random, add it to MIS and its neighbor to MVC and delete these two nodes from the graph.
- If there is no vertex of degree one, then we identify the node(s) with maximum degree, search all the neighbors of this node(s) and add the node with minimum degree to MIS and its neighbors to the MVC. Afterwards, we delete the selected node and all its neighbors from the graph.

The above process is repeated till no vertex remains in the input graph. The pseudocode for the proposed algorithm is presented below.

Pseudocode of the proposed algorithm. Procedure. VCUMI (Graph, Max_nodes)

This procedure will select MVC nodes by finding and deleting all MIS nodes. MVC and MIS are two disjoint sets, which contain the nodes selected under these categories. The graph at each stage is defined by two sets, Edge, which is a list of the edges, and *V*, which is a list of the nodes in this graph. Maxdeg is the set containing a list of the nodes of maximum degree at the present step of the algorithm. Adjacent is the set containing all the nodes adjacent to at least one node in the current maxdeg set. The MIS and MVC are constructed step by step.

```
[Repeat the following process till no edge remains in the graph]
While (Edge is not empty)
    For each (v ∈ V)
        Find the degree of v
    [End of for loop]
    [When multiple nodes of degree one exist, select one of them at random]
If  There is/are node(s) of degree one, then
        If there is a single node of degree one, then
            Add it to MIS and add its neighbor to MVC.
            Remove these nodes from the present graph.
Else if  There is more than one node of degree one, then
            Select one of these nodes at random and add it to MIS and its
            neighbor to MVC.
            Remove these nodes from the present graph
```

[End inner if clause]
Else:
The 'maxdeg' set is defined to consist of all the nodes of maximum degree in the present graph.
The 'adjacent' set is defined to consist of all the nodes in the present graph adjacent to at least one node in the set 'maxdeg'.
Find the node of minimum degree in the adjacent set, add this node to 'MIS',
Add all the neighbors of this node to 'MVC',
Remove this minimum degree node, its neighbors from the present graph
[End of if statements]
[End of while loop]
Return.

The primary aim of the design of this algorithm was to make it as simple as possible. Hence, we have not included any complex steps for making decisions about vertex selection. However, this does not negatively affect either the functioning of the algorithm or the quality of the solution. If we analyze the pseudo code for the proposed algorithm, then it is clear that the whole process will be repeated at most $|V|/2$ times, where $|S|$ denotes the cardinality of the set S (at each step at least two nodes are removed from the graph) and the internal loop for calculating the degree will also be repeated at most $|V|$ times, so the total complexity in the worst case will be $O(|V|^2)$. In practical experiments, we have shown that the algorithm can solve most of the benchmark problems in a fraction of the time taken by other algorithms that have either the same or slightly higher worst case complexity.

4. Empirical results

We tried to collect the best available benchmarks, in order to reflect the practical efficiency of the proposed algorithm in comparison to other approaches. All the results obtained using these algorithms are reported here without any refinement or enrichment. The algorithms that were selected for comparison are maximum degree greedy (MDG) [5] and the near optimal vertex cover approximation (NOVAC) [15]. These algorithms were chosen on the basis of similarities in their working mechanisms to our proposed approach. MDG and NOVAC work on the same greedy approach with differences in the procedure for selecting a vertex to include in the MVC. Our proposed algorithm VCUMI is also based on a greedy approach with a different mechanism for processing a graph. MDG is one of the earliest greedy approaches proposed for MVC,

where the candidates for addition into the MVC are the vertices of maximum degree in the current graph. NOVAC extracts vertices that are potential nodes for MVC by finding the nodes that are adjacent to the minimum degree nodes. All of these algorithms were implemented in MATLAB and tested using the same platform.

To evaluate the efficiency of the proposed algorithm, many experiments were performed on randomly generated graphs for which the optimal minimal vertex cover is known. These graphs were obtained using a process outlined by Sanchis [21]. The remaining benchmark problems were collected from [23] and are frequently used by researchers of this field [2, 16, 10, 17, 18]. These benchmark problems include information about the total number of nodes and the optimal solution, which helps researchers to assess the results of their proposed approaches against some know dataset.

Table 1 provides the results obtained using our proposed algorithm. The first column indicates the name of the benchmark problem. The second and third columns provide information about the total number of vertices in the benchmark problem and the number of nodes in the MVC, respectively. The fourth column provides information about the number of nodes selected using our approach and the fifth column provides the approximation ratio.

Table 1. Benchmark results for VCUMI, MDG and NOVAC-1

No.	Benchmark problem	No. of vertices	Optimal MVC	VCUMI	MDG	NOVAC-1	Approximation ratio		
							VCUMI	MDG	NOVAC-1
1	graph50-01	50	30	30	30	30	1	1	1
2	graph50-02	50	30	30	30	30	1	1	1
3	graph50-03	50	30	30	30	30	1	1	1
4	graph50-04	50	40	40	41	41	1	1.025	1.025
5	graph50-05	50	27	27	27	27	1	1	1
6	graph50-06	50	38	38	38	38	1	1	1
7	graph50-07	50	35	35	35	35	1	1	1
8	graph50-08	50	29	29	29	29	1	1	1
9	graph50-09	50	40	40	40	40	1	1	1
10	graph50-10	50	35	35	35	35	1	1	1
11	graph100-01	100	60	60	60	60	1	1	1
12	graph100-02	100	65	65	65	65	1	1	1
13	graph100-03	100	75	75	75	75	1	1	1
14	graph100-04	100	60	60	60	60	1	1	1
15	graph100-05	100	60	60	60	60	1	1	1
16	graph100-06	100	80	80	80	80	1	1	1
17	graph100-07	100	65	65	65	65	1	1	1
18	graph100-08	100	75	75	75	75	1	1	1
19	graph100-09	100	85	85	85	85	1	1	1
20	graph100-10	100	70	70	70	70	1	1	1
21	graph200-01	200	150	150	150	150	1	1	1
22	graph200-02	200	125	125	125	125	1	1	1

No.	Benchmark problem	No. of vertices	Optimal MVC	VCUMI	MDG	NOVAC-1	Approximation ratio		
							VCUMI	MDG	NOVAC-1
23	graph200-03	200	175	175	175	175	1	1	1
24	graph200-04	200	140	140	140	140	1	1	1
25	graph200-05	200	150	150	150	150	1	1	1
26	graph500-01	500	350	350	350	350	1	1	1
27	graph500-02	500	400	400	400	400	1	1	1
28	graph500-03	500	375	375	375	375	1	1	1
29	graph500-04	500	300	300	300	300	1	1	1
30	graph500-05	500	290	290	290	290	1	1	1
31	phat300-1	300	292	293	293	293	1.0034	1.0034	1.0034
32	phat300-2	300	275	278	278	275	1.0109	1.0109	1
33	phat300-3	300	264	268	269	266	1.0152	1.0189	1.0076
34	phat700-1	700	689	692	693	692	1.0044	1.0058	1.0044
35	phat700-2	700	656	660	660	657	1.0061	1.0061	1.0015
36	phat700-3	700	638	640	642	641	1.0031	1.0063	1.0047
37	jhonson8-2-4	28	24	24	24	24	1	1	1
38	jhonson8-4-4	70	56	56	62	56	1	1.1071	1
39	jhonson16-2-4	120	112	112	112	112	1	1	1
40	jhonson32-2-4	496	480	480	480	480	1	1	1
41	sanr200-0.7	200	182	184	184	185	1.011	1.011	1.0165
42	sanr200-0.9	200	158	161	164	159	1.019	1.038	1.0063
43	sanr400-0.5	400	387	389	392	388	1.0052	1.0129	1.0026
44	sanr400-0.7	400	379	383	384	381	1.0106	1.0132	1.0053
45	fbr-30-15-5	450	420	424	429	424	1.0095	1.0214	1.0095
46	fbr-35-17-2	595	560	568	570	565	1.0143	1.0179	1.0089
47	c 125	125	91	94	93	92	1.033	1.022	1.011
48	c 250	250	206	211	211	211	1.0243	1.0243	1.0243
49	c500.9	500	≤443	450	453	449	1.0158	1.0226	1.0135
50	broc200-2	200	188	191	192	190	1.016	1.0213	1.0106
51	gen200-p0.9-44	200	156	158	165	163	1.0128	1.0577	1.0449
52	Hamming6-2	64	32	32	32	32	1	1	1
53	Hamming6-4	64	60	60	60	60	1	1	1
54	Hamming8-2	256	128	128	128	128	1	1	1
55	Hamming8-4	256	240	240	248	240	1	1.0333	1
56	Hamming10-2	1024	512	512	512	512	1	1	1
57	dsjc500	500	487	488	491	488	1.0021	1.0082	1.0021
58	keller4	171	160	162	164	164	1.0125	1.025	1.025
59	keller5	776	749	754	764	761	1.0067	1.02	1.016
60	cfat200-1	200	188	188	188	188	1	1	1
61	cfat200-2	200	176	176	176	176	1	1	1
62	cfat200-5	200	142	142	142	142	1	1	1
63	cfat500-1	500	486	486	486	486	1	1	1
64	cfat500-2	500	474	474	474	474	1	1	1
65	cfat500-5	500	436	436	436	436	1	1	1
66	mann-a27	378	252	253	261	253	1.004	1.0357	1.004

It is evident from Table 1 that VCUMI was able to find the optimal solution to almost 68% of the benchmark problems, while MDG and NOVAC found the optimal solution to 63% and 68% of the benchmark problems, respectively. A more complete comparative analysis of these results shows that VCUMI, i.e. the proposed algorithm, outperforms the other techniques in producing quality solutions. Based on the whole range of these benchmark graphs, VCUMI gave a maximum error ratio of 1.033, while MDG and NOVAC-1 gave maximum error ratios of 1.107 and 1.04, respectively. Based on these worst error ratios, it is evident that VCUMI is always able to produce efficient solutions and, in terms of worst case performance, VCUMI produces more accurate results than the other algorithms. The average error ratios were also calculated. For VCUMI, the average error ratio is 1.004 and the average error ratios for MDG and NOVAC-1 are equal to1.009 and 1.004, respectively. Results for the running time complexity of these algorithms also suggest that VCUMI on average performs better than the other algorithms. Computationally, our proposed algorithm outperforms the others, because on average more than one node is added to the MVC at each step. Figure 3 compares the approximation ratios of the three algorithms tested.

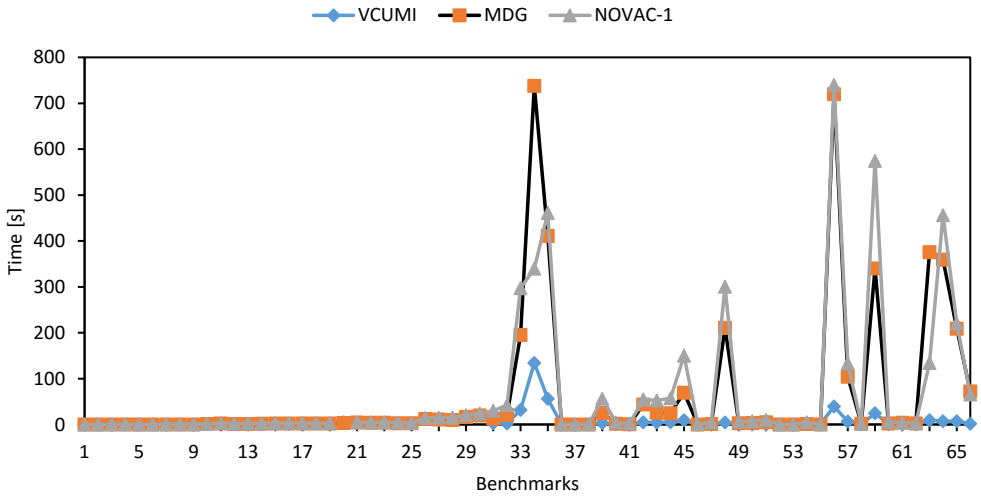

Fig. 3. Approximation ratios of VCUMI, MDG, and NOVAC-1

During this test procedure, we noted one critical point that can help other researchers in designing approximation algorithms for NP Complete problems. Namely, if the nature of a problem is complex, then try an alternative approach and perform a reduction. Such an alternative approach often gives a very fast solution and thus VCUMI is one of the fastest algorithms for minimum vertex cover. Figure 4 illustrates the time required compared to that of other algorithms.

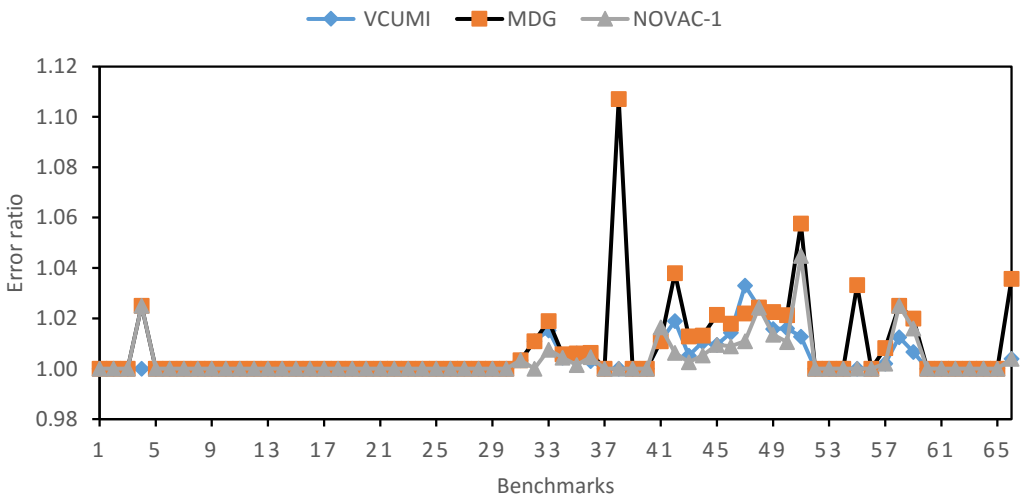

Fig. 4. Run time for VCUMI, MDG and NOVAC-1

5. Conclusion

A new approximation algorithm, VCUMI (vertex cover using maximum independent set) which is based on a greedy approach, has been proposed. Experiments carried out on numerous benchmark problems indicate the efficiency of the proposed algorithm in terms of both error ratio and the time required to solve a problem. After testing our approach using standard benchmark problems, the proposed algorithm gave a maximum error ratio of 1.033 with an average error rate of 1.004. Both the average and maximum errors are lower than for the two other competing algorithms that use a similar strategy.

If we consider the computational time, then it should be obvious that it is difficult to outperform the maximum degree greedy (MDG) algorithm, due to its inherent simplicity (it does not require any complex sub-procedure). However, in terms of computation time, our algorithm clearly performs better than NOVAC and even outperformed MDG for the majority of the benchmark problem.

We have solved the problem of finding a MVC for graphs by making use of the MIS rather than directly finding the MVC. The results obtained using this proposed approach were more accurate and the algorithm itself far more efficient than other algorithms that solve MVC directly.

Acknowledgement

We are thankful to our friends who helped us a lot while working on this research. We are also thankful to those researchers who have shared the contents of their work on the Internet.

References

[1] ASGEIRSSON E.I., STEIN C., *Divide-and-conquer approximation algorithm for vertex cover*, SIAM Journal on Discrete Mathematics, 2009, 23 (3), 1261.
[2] BALAJI S., SWAMINATHAN V., KANNAN K., *Optimization of unweighted minimum vertex cover*, World Academy of Science, Engineering and Technology, 2010, 43, 716.
[3] CHEN J., HUANG X., KANJ I.A., XIA G., *Linear FPT reductions and computational lower bounds*, Proc. 36th Annual ACM Symposium on Theory of Computing, ACM, 2004, 212.
[4] CHVATAL V., *A greedy heuristic for the set-covering problem*, Mathematics of Operations Research, 1979, 4 (3), 233.
[5] CLARKSON K.L., *A modification of the greedy algorithm for vertex cover*, Information Processing Letters, 1983, 16 (1), 23.
[6] CORMEN T.H., LEISERSON C.E., RIVEST R.L., STEIN C., *Introduction to Algorithms*, 2, MIT Press, Cambridge 2001.
[7] AVIS D., IMAMURA T., *A list heuristic for vertex cover*, Operations Research Letters, 2007, 35 (2), 201.
[8] DA SILVA M.O., GIMENEZ-LUGO G.A., DA SILVA M.V., *Vertex cover in complex networks*, International Journal of Modern Physics C, 2013, 24 (11).
[9] DELBOT F., LAFOREST C., *A better list heuristic for vertex cover*, Information Processing Letters, 2008, 107 (3), 125.
[10] DELBOT F., LAFOREST C., *Analytical and experimental comparison of six algorithms for the vertex cover problem*, Journal of Experimental Algorithmics, 2010, 15, 1.
[11] DEMAINE E.D., FOMIN F.V., HAJIAGHAYI M., THILIKOS D.M., *Subexponential parameterized algorithms on bounded-genus graphs and h-minor-free graphs*, Journal of the ACM, 2005, 52 (6), 866.
[12] DINUR I., SAFRA S., *The importance of being biased*, Proc. 34th Annual ACM Symposium on Theory of Computing, ACM, 2002, 33.
[13] DINUR I., SAFRA S., *On the hardness of approximating minimum vertex cover*, Annals of Mathematics, 2005, 162 (1), 439.
[14] GAREY M.R., JOHNSON D.S., *Computers and Intractability. A Guide to the Theory of NP--Completeness*, W.H. Freeman & Co., New York 1979, 29.
[15] GAJUREL S., BIELEFELD R., *A simple NOVAC. Near optimal vertex cover algorithm*, Procedia Computer Science, 2012, 9, 747.
[16] HALLDORSSON M.M., RADHAKRISHNAN J., *Greed is good: Approximating independent sets in sparse and bounded-degree graphs*, Algorithmica, 1997, 18 (1), 145.
[17] IMRAN K., HASHAM K., *Modified vertex support algorithm. A new approach for approximation of minimum vertex cover*, Research Journal of Computer and Information Technology Sciences, 2013, 1 (6), 7.
[18] IMRAN K., ISRAR A., MUZAMMIL K., *AVSA, modified vertex support algorithm for approximation of MVC*, International Journal of Advanced Science and Technology, 2014, 67, 71.
[19] KARP R.M., *Reducibility among Combinatorial Problems*, Springer, 1972.
[20] LI S., WANG J., CHEN J., WANG Z., *An approximation algorithm for minimum vertex cover on general graphs*, The Third International Symposium on Electronic Commerce and Security Workshops, ISECS, Guangzhou, P.R. China, 2010, 249.

[21] SANCHIS L.A., *Test case construction for the vertex cover problem*, Series in Discrete Mathematics and Theoretical Computer Science, DIMACS, 1994, 15, 315.
[22] SINGH A., GUPTA A.K., *A hybrid heuristic for the minimum weight vertex cover problem*, Asia--Pacific Journal of Operational Research, 2006, 23 (2), 273.
[23] *Vertex Cover Benchmark Instances*, http://www.cs.hbg.psu.edu/txn131/vertex_cover.html, retrieved April 23, 2015.
[24] XU X., MA J., *An efficient simulated annealing algorithm for the minimum vertex cover problem*, Neuro-Computing, 2006, 69 (7), 913.

Received 23 December 2014
Accepted 30 November 2015