NMVSA Greedy Solution for Vertex Cover Problem

Mohammed Eshtay
Department of Computer Science
Applied Science Private University
Amman, Jordan

Azzam Sleit
Department of Computer Science
University of Jordan
Amman, Jordan

Ahmad Sharieh
Department of Computer Science
University of Jordan
Amman, Jordan

***Abstract*—Minimum vertex cover (MVC) is a well-known NP-Complete optimization problem. The importance of MVC in theory and practical comes from the wide range of its applications. This paper describes a polynomial time greedy algorithm to find near optimal solutions for MVC. The new algorithm NMVAS is a modification of already existed algorithm called MVAS which uses the same principle of selecting candidate from the neighborhood of the vertex with a modification in the selection procedure. A comparative study is conducted between the NMVAS and MVAS which shows that the proposed algorithm NMVSA provides better or equal results in the most cases of the underlying data sets which leads to a better average approximation ratio of NMVAS. NMVAS inherits the simplicity of the original algorithm.**

Keywords—Vertex Cover Problem (MVC); Combinatorial Problem; NP-Complete Problem; Approximation Algorithm; Greedy algorithms; Minimum Independent Set

I. INTRODUCTION

Many problems and concrete situations (networks, vehicle routing, maps, etc.) are mapped into graphs and then set of algorithms are developed to manipulate these graphs to reach a solution for the given problem. Unfortunately many of such problems cannot be solved exactly in a polynomial time. Such problems are called intractable but because of their importance they must be solved.[10]

One option to solve such NP-Problems is to use heuristics. Heuristics do not guarantee the quality of solution and no guarantee also on the time required to solve the problem. Another alternative is to use approximation algorithms. In approximation algorithms we find a solution with quality between optimal and r (r>=1) and we can guarantee that the algorithm will finish running in a reasonable time.

The problem that we are concern about in this paper is MVC. MVC the minimum vertex cover is a popular NP-Complete problem with a high importance in theory and practical computer science.[1].

There are many algorithms proposed to solve this problem. In this paper we will discuss set of solutions existed in the literature to solve this problem using approximation algorithms. In addition we will present new modified algorithm NMVSA and compare the results with already existed solutions of the same type.

Vertex cover is equivalent to two other popular optimization problems: MC (The Maximum Clique Problem) and MIS (Minimum Independent Set). These three problems actually can be considered as three different forms of the same problem. [5,19,20]

The following propositions hold for undirected graph G(V,E)

- The minimum vertex cover does not contain vertex with degree zero.
- Every vertex cover contains v or w where w belongs to set of vertices adjacent to v, for every vertex v.
- If there is a vertex v of degree one, w is adjacent to u, there is a minimum vertex cover which contains w.
- If there is a vertex v of degree two, {u,w} are adjacent to u then there is a minimum vertex cover which contains both u and w or neither of them. [5,19]

Definition: A vertex cover in an undirected graph G is the subset of vertices S in such that every edge in the graph G is connected to at least one vertex of S. The size of a vertex cover is the number of vertices it contains. [1]

Vertex cover takes undirected graph G as input and integer number S. It tries to find a vertex cover of size S.

The size of the vertex cover is the number of vertices inside it. If the nodes are weighted by a non-negative number then it tries to find vertex cover with the minimum weighted vertices [2,15].

The vertex cover has applications in bio-informatics, security and networking. An example of vertex cover problem could be forming a team to perform certain set of tasks then we must hire enough people to accomplish each certain task. Many applications can use MVC such as network security, scheduling, biology and finance [3,16].

There are exact solutions for this problem that guarantee the optimal solution such as branch and bound but may fail to give solution within reasonable time for large instances. Other type of solution is to use heuristic algorithms. Heuristic algorithms do not guarantee the optimal solutions but they can find optimal or near optimal solution in a reasonable time [4,21,22].

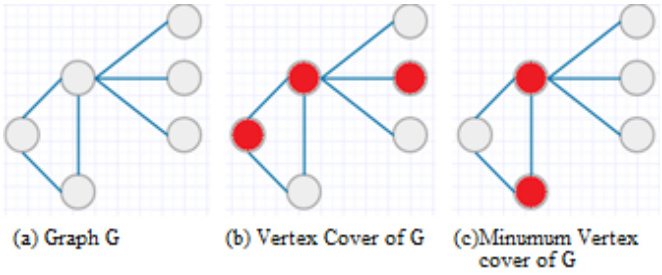

Fig. 1. Vertex Cover

Notations

Given an undirected graph G(E,V), a vertex cover of a graph G is a subset of vertices VC $\subseteq$ V(G) such that every edge has an end point in VC. (That is, for all e $\in$ E(G), e $\cap$ VC $\neq \phi$).

Fig. 1 shows examples of a vertex cover and a minimum vector cover respectively. The nodes colored red in Fig. 1b are the nodes selected in the vector cover. The minimum vertex cover is a vertex cover of smallest possible size that appear in Fig. 1c.

The remainder of this paper is organized as follows, the different methods and related work in Section 2, the terminology, algorithm and the computation complexity in Section 3. We then discuss and analyze the results of experiments in section 4. Section 5 contains the conclusions and points to several directions for further research.

II. Related Work

The NP-Completeness of MVC problem has proved by R Karp[6]. This means that we don't have a polynomial time solution to solve it. The problem is one of the most well studied problems from researchers in computer science due to the importance and wide-range of its applications. As I mentioned earlier we can use approximation algorithms such as APPROX-VERTEX-COVER and Maximum Degree Greedy (MDG) to solve the problem.[7]

The Depth First Search algorithm presented by savage [8] computes the spanning tree at the beginning and then returns the non-leaves vertices of the depth first search spanning tree as a vertex cover.

The Maximum Degree Greedy (MDG) algorithm is an adaptation of a well-known heuristic algorithm used for extracting the MIS MDG keeps adding the node with the maximum degree until all edges in the graph are covered. [7,9,10]

The Greedy Independent Cover (GIC) is an adaptation of the algorithm presented in. In this algorithm we select the vertex of the minimum degree and all its neighbors to the vertex cover, the process continues until we cover all edges. [10,11]

In the algorithm LISTLEFT is proposed to find vertex cover based on a list heuristic. In this algorithm the selection of the vertices is known in advance and cannot be changes. Another list heuristic algorithm ListRight is proposed in [12].

Many of the previous mentioned methods to solve the problem of MVC depend on the degree of the vertex itself. Balaji et al. presented another technique that depends of a value called the support of the vertex. They proposed algorithm called vertex support algorithm (VSA). [6,13]

A modification of the VAS is called MVAS where the selection of the vertex does not depend only on the vertex that have the maximum the support value but it finds all the vertices with minimum support value and then it selects the vertex with minimum support from the list of all neighbors of the selected vertices. MVAS shows better results in some benchmarks comparing to original VAS. [6, 13]

Some other solutions depend on genetic algorithms such as (HVX). Xu and Ma presented solution that uses annealing algorithms to find the minimum vertex cover. In their work they show almost 100% approximation ratio for some benchmarks but they need to apply it on more benchmarks. [14].

In this paper we are introducing new greedy algorithm (NMVAS) to find the MVC by modifying MVAS. The results of applying the two algorithms in a set of benchmark sets are then compared.

III. Terminologies, Algorithm and Computational Complexity

In this section we present the proposed algorithm, the pseudo code of the algorithm, example to clarify the idea, and theoretical complexity analysis.

A. Terminology

The selection of vertices in NMVSA algorithm relies on the degree of the neighborhood vertices. We will introduce the definition of neighborhood, degree, and support of vertex:

Neighborhood of a vertex: Let G be an undirected graph G (V, E) where V is set of vertices and E is set of Edges. |E|=m, |V|=n. For each v$\in$V the neighborhood of v-N (v) = {u$\in$V | u is adjacent to v.

Degree of a vertex: The degree of the vertex d (v) is the number of adjacent neighbors for vertex v $\in$ V.

The support of a vertex: support of a vertex s (v) is the sum of degrees of all neighbors of v.

$$s(v) = \sum_{u \in N(v)} d(u)$$

Vertex Cover: Vertex cover c={x $\in$ V | x=u or v if (v,u) is an edge e$\in$E}

B. The proposed Algorithm

In the proposed algorithm we are trying to find an optimal or near optimal solution of the vertex cover problem because of the absence of a polynomial time solution for this NP-Complete problem. The proposed solution depends on the idea presented in VAS algorithm [17]. The selections of the vertices that will be part of the vertex cover rely on the value of support. Value of support is a value represents the sum of the degrees of the neighbors of the vertices.

The algorithm starts by finding the degree of each vertex not yet selected in the vertex cover. The degree of the vertex is the number of adjacent neighbors for vertex. The second stage after finding the degrees is finding the support value for each vertex. The algorithm proceeds by finding a list that contains all vertices that have the minimum support value. The next step is to select the vertex with the maximum support value among the neighbors of the vertices of the minimum support value. After adding the vertex to the vertex cover, all adjacent edges to this vertex are deleted. The process continues until no more edges exist.

C. New Modified Vertex Support Algorithm (NMVSA)

The idea of selection in the proposed algorithm NMVSA shown in the algorithm 1 depends on the fact that the candidates of vertex cover are adjacent to the vertices with minimum degrees. [18] NMVSA adds a vertex from the supporting list with the maximum degree in each iteration to the vertex cover and delete all edges connected to this vertex. The process continues until no more edges still in E.

Algorithm 1: NMVSA

Input: $G(V, E)$.
Output: Vc
1: **While** ($E \neq \emptyset$)
//Calculate the degree for each vertex
2: **Foreach** $v_i \in V$
Find d (v_i)
3: minSup = ∞
//Calculate the support value for each vertex
4: **Foreach** $v_i \in V$
Calculate the support of $v_i \leftarrow s(v_i)$
If $s(v_i)$ <minSup Then
minSup = $s(v_i)$
//Create the list of minimum support vertices
5: **Foreach** $v_i \in V$
If $s(v_i) =$ minSup Then
Add v_i to minSupList L
//Find the neighbors of minimum support vertices
6: **Foreach** $v_i \in L$
Add $N(v_i)$ to neighborhoodList H
maxSupNeighbor = $-\infty$
c = 0 //Candidate vertex
//Find the vertex to be added to the Vc
7: **Foreach** $v_i \in H$
If $s(v_i) >$ maxSupNeighbor Then
c= (v_i)
maxSupNeighbor = $s(v_i)$
$Vc = Vc \cup c$
$V = V - c$
$E = E - \{e \in E \mid e = (c, u), u \in V\}$

Step one guarantees that we pass through all edge, at each iteration we calculate the degrees of all vertices in step 2. In step 3,4 we find the support value for each vertex and we find the minimum support value among all vertices. Step 5 finds all the vertices that have the minimum support value. In step 6 we find all the neighbors of vertices listed in the minimum support vertices list. In step 7 we select the vertex with the maximum support value from the list of all neighbors of the vertices with minimum support value.

The intuition behind the algorithm is to select the vertices that connect as much as possible from the vertices that are located on the edges of the graph.

D. Computational Complexity

According to our terminology, number of vertices is n and number of edges is m. The complexity of NMVSA can be obtained as follows: step 2 in the algorithm which is used to find the degree of each vertex requires$O(n^2)$. Step 4 which is used to calculate the support value of each vertex and finding the minimum support value is taking $O(n^2)$ also. To pick all the vertices that have supporting value equal to the minimum support value in step 5 we need $O(n)$. Step 7 requires$O(n)$. The whole process will be repeated m times in the worst case.

The total running time can be expressed in the following formula:

$$m.[O(n^2) + O(n^2) + O(n) + O(n)] = O(mn^2)$$

IV. Experimental Results and Analysis

The code of NMVSA has been developed using c++ on a machine Intel Core 2 Duo 2.1 GHz with 2GB memory. The algorithm has been compared to three other algorithms - VSA[17], MVSA[6], and modified MDG[2].

The programs have been executed on part of well-known dataset DIMACS benchmark set and on BHOSLIB benchmark set. The effectiveness of the algorithm NMVSA algorithm was evaluated by executing the algorithm on 40 instances of BHOSLIB 13 instances of DIMACS dataset.

Table I lists the results of all experiments hold in the algorithms. Table I consists of two parts. The first part is the length of the vertex cover found by each of the algorithms. The second part shows the ratio of the result of each algorithm and the best results obtained. The first column of the table states the name of the instance, the second column is the cardinality of the instance, and the following two columns are the results of applying MVSA and our proposed algorithm NMVSA respectively.

The second part of the table is the Approximation ratio of each of the tested algorithms.

According to Table I we can see that NMVSA gives better or equal results in all instances that we used from DIMACS benchmark. also NMVSA gives better results in 4 instances of BHOSLIB benchmark set and not as good as MVSA in two instance of BHOSLIB.

TABLE II. EXPERIMENTAL RESULTS

Benchmark	|V|	Optimal Vc	MVAS	NEW MVAS	Approximation ratio	
					MVAS	NEW MVAS
frb30-15-1	450	420	426	426	1.014	1.014
frb30-15-2	450	420	427	427	1.017	1.017
frb30-15-3	450	420	426	426	1.014	1.014
frb30-15-4	450	420	426	426	1.014	1.014
frb30-15-5	450	420	429	428	1.021	1.019
frb35-17-1	595	560	567	567	1.013	1.013
frb35-17-2	595	560	565	565	1.009	1.009
frb35-17-3	595	560	567	567	1.013	1.013
frb35-17-4	595	560	567	567	1.013	1.013
frb35-17-5	595	560	567	567	1.013	1.013
frb40-19-1	760	720	728	728	1.011	1.011
frb40-19-2	760	720	728	728	1.011	1.011
frb40-19-3	760	720	728	728	1.011	1.011
frb40-19-4	760	720	729	730	1.013	1.014
frb40-19-5	760	720	728	728	1.011	1.011
frb45-21-1	945	900	910	910	1.011	1.011
frb45-21-2	945	900	909	909	1.01	1.01
frb45-21-3	945	900	908	908	1.009	1.009
frb45-21-4	945	900	909	909	1.01	1.01
frb45-21-5	945	900	910	910	1.011	1.011
frb50-23-1	1150	1100	1111	1111	1.01	1.01
frb50-23-2	1150	1100	1111	1111	1.01	1.01
frb50-23-3	1150	1100	1109	1109	1.008	1.008
frb50-23-4	1150	1100	1111	1111	1.01	1.01
frb50-23-5	1150	1100	1112	1112	1.011	1.011
frb53-24-1	1272	1219	1229	1229	1.008	1.008
frb53-24-2	1272	1219	1229	1229	1.008	1.008
frb53-24-3	1272	1219	1230	1230	1.009	1.009
frb53-24-4	1272	1219	1230	1230	1.009	1.009
frb53-24-5	1272	1219	1230	1230	1.009	1.009
frb56-25-1	1400	1344	1355	1355	1.008	1.008
frb56-25-2	1400	1344	1353	1355	1.007	1.008
frb56-25-3	1400	1344	1355	1355	1.008	1.008
frb56-25-4	1400	1344	1357	1357	1.01	1.01
frb56-25-5	1400	1344	1354	1353	1.007	1.007
frb59-26-1	1534	1475	1487	1486	1.008	1.007
frb59-26-2	1534	1475	1487	1487	1.008	1.008
frb59-26-3	1534	1475	1483	1483	1.005	1.005
frb59-26-4	1534	1475	1487	1485	1.008	1.007
frb59-26-5	1534	1475	1487	1487	1.008	1.008
C500.9	500	443	451	451	1.018	1.018
C1000.9	1000	932	945	945	1.014	1.014
C2000.5	2000	1984	1988	1988	1.002	1.002
C2000.9	2000	1920	1934	1933	1.007	1.007
keller4	171	160	160	160	1	1
keller5	776	749	752	751	1.004	1.003
keller6	3361	3302	3311	3311	1.003	1.003
p_hat300-1	300	292	294	294	1.007	1.007
p_hat300-2	300	275	279	279	1.015	1.015
p_hat300-3	300	264	272	271	1.03	1.027
p_hat700-1	700	689	692	692	1.004	1.004
p_hat700-2	700	656	660	660	1.006	1.006
p_hat700-3	700	638	649	649	1.017	1.017

Table II summarizes the results of the execution of the algorithm. Table II shows that our modified version of MVSA gave equal results in 40 instances, better solutions in 7 instances and MVSA was better in two instances.

TABLE III. SUMMARY OF RESULTS

	NMVAS
Better	7
Equal	40
Worse	2

V. CONCLUSION

In this paper we developed new algorithm called NMVSA by modifying existing algorithm called MVSA. We conducted a comparison study depending on the results of execution on two different well known benchmark sets. NMVSA gives optimal or near optimal solutions with average ratio 1.0101on the instances that we executed the algorithm in. NMVSA generally shows better results than MVSA algorithm.

Future work includes trying to get more optimal solutions and decrease the ratio to be closer to 1

ACKNOWLEDGMENT

The authors are grateful to the Applied Science Private University, Amman, Jordan for the full financial support granted to this research project (Grant No. DRGS-2015-2016-54).

REFERENCES

[1] Cormen, T. H. (2013). Algorithms unlocked. Mit Press.

[2] Cai, S., Su, K., Luo, C., & Sattar, A. (2013). NuMVC: An efficient local search algorithm for minimum vertex cover. Journal of Artificial Intelligence Research, 687-716.

[3] Angel, E., Campigotto, R., & Laforest, C. (2012). Implementation and comparison of heuristics for the vertex cover problem on huge graphs. In Experimental Algorithms (pp. 39-50). Springer Berlin Heidelberg.

[4] Williamson, D. P., & Shmoys, D. B. (2011). The design of approximation algorithms. Cambridge university press.

[5] Chandran, L. S., & Grandoni, F. (2005). Refined memorization for vertex cover. Information Processing Letters, 93(3), 125-131.

[6] 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 ISSN, 2320, 6527.

[7] Bansal, S., & Rana, A. Analysis of Various Algorithms to Solve Vertex Cover Problem.

[8] Savage, C. (1982). Depth-first search and the vertex cover problem. Information Processing Letters, 14(5), 233-235.

[9] Bodlaender, H. L., Thilikos, D. M., & Yamazaki, K. (1997). It is hard to know when greedy is good for finding independent sets. Information Processing Letters, 61(2), 101-106.

[10] Delbot, F., & Laforest, C. (2010). Analytical and experimental comparison of six algorithms for the vertex cover problem. Journal of Experimental Algorithmics (JEA), 15, 1-4.

[11] Halldórsson, M. M., & Radhakrishnan, J. (1997). Greed is good: Approximating independent sets in sparse and bounded-degree graphs. Algorithmica, 18(1), 145-163.

[12] Delbot, F., & Laforest, C. (2008). A better list heuristic for vertex cover. Information Processing Letters, 107(3), 125-127.

[13] Khan, I., & Khan, S. (2014). Experimental Comparison of Five Approximation Algorithms for Minimum Vertex Cover. International Journal of u-and e-Service, Science and Technology, 7(6), 69-84.

[14] Xu, X., & Ma, J. (2006). An efficient simulated annealing algorithm for the minimum vertex cover problem. Neurocomputing, 69(7), 913-916.

[15] Hammo, B., Sleit, A., & El-Haj, M. (2007). Effectiveness of query expansion in searching the Holy Quran.

[16] Sleit, A., Abusharkh, S., Etoom, R., & Khero, Y. (2012). An enhanced semi-blind DWT–SVD-based watermarking technique for digital images. The Imaging Science Journal, 60(1), 29-38.

[17] Balaji, S., Swaminathan, V., & Kannan, K. (2010). Optimization of unweighted minimum vertex cover. World Academy of Science, Engineering and Technology, 43, 716-729.

[18] Gajurel, S., & Bielefeld, R. (2012). A fast near optimal vertex cover algorithm (novca). IJEA, 3(1), 9-18.

[19] Sharieh, A., Al Rawagepfeh, W., Mahafzah, M., & Al Dahamsheh, A. (2008). An algorithm for finding maximum independent set in a graph. European Journal of Scientific Research, 23(4), 586-596.

[20] Al-Jaber, & A., Sharieh, A. (2000). Algorithms based on weight factors for maximum independent set,DIRASAT, Nat. Sci.,27(1), 74–90,.

[21] Sleit, A., Serhan, S., & Nemir, L. (2008, August). A histogram based speaker identification technique. In Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008. First International Conference on the (pp. 384-388). IEEE.

[22] Sleit, A., Qatawneh, M., Al-Sharief, M., Al-Jabaly, R., & Karajeh, O. (2011). Image Clustering using Color, Texture and Shape Features. KSII Transactions on Internet & Information Systems, 5(1).