Research Journal of Computer and Information Technology Sciences ISSN 2320 – 6527
Vol. **1(6),** 7-11, November **(2013)** *Res. J. Computer and IT Sci.*

Modified Vertex Support Algorithm: A New approach for approximation of Minimum vertex cover

Khan Imran[1] and Khan Hasham[2]
[1]University of Swat, PAKISTAN
[2]SZABIST, Islamabad, PAKISTAN

Available online at: www.isca.in, www.isca.me
Received 14th October 2013, revised 2nd November 2013, accepted 18th November 2013

Abstract

Graph related problems mostly belong to NP class and minimum vertex cover is one of them. Minimum vertex cover is focus point for researchers since last decade due to its vast areas of application. In this research paper we have presented a modified form of approximation algorithm for minimum vertex cover which makes use of data structure proposed already named vertex support. We changed the way of selection slightly from vertex support algorithm, vertices attached to minimum support node play very critical role in selection of vertices for minimum vertex cover and we used this in our algorithm. Using our approach we managed to reduce worst case approximation ratio of VSA which is 1.583 to 1.064, this is very major change in providing results with simplicity. Results are also compared with MDG and NOVAC in order to demonstrate the efficiency of selecting vertices in this manner. Simplicity in design can help in applying it in time restricted environments.

Keywords: MVC (Minimum vertex cover), MIS (Maximum independent sets), DCA (Degree Contribution Algorithm), VSA (Vertex Support Algorithm), DC (Degree Contribution), MWVC (Minimal Weighted Vertex Cover), MWIS (Maximal Weighted Independent Set).

Introduction

Graph G (V, E) is a combination of vertices and edges. Vertices or nodes are connected through edges. Many real life problems can be modeled using graphs and after modeling, these problems are manipulated by several techniques to optimize the specific objective of the area of the application. Application areas of MVC include wireless communications, civil, electrical engineering, multiple alignment of biochemistry, Bioinformatics etc[1]. A problem with graph theory is that many problems are intractable i.e., these cannot be solved in polynomial time and majority of the graph related problems belong to a class called NP-Complete. As it is widely believed that NP-Complete problems cannot be solved optimally in polynomial time, various alternative approaches have been considered by the researchers to solve these problems. These techniques are either based on some complex heuristics or approximation of the optimal actual solution. Heuristic solutions have no guarantee of producing a quality solution in reasonable amount of time in many cases. 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 depends on the approximation ratio. Approximation ratio is defined as the ratio of approximate solution to the actual optimal solution, $\rho i = A(i)/OPT(i) \geq 1$, where 'i' is an instance of the problem, 'A' is the approximate solution and OPT is the optimal solution. ρi is the approximation ratio for a particular problem instance 'i' and $\rho n = \text{Max}_i\rho i$ for all n, i.e. ρn is the maximum value of all ρis When ρi=1 then the approximate solution is actual optimal solution, but this is not the case always because these problems are intractable and according to Garey and Jhonson a problem is intractable if it is so hard that no polynomial algorithm can possibly solve it[2]. The value of ρi determines the quality of solution, the more it deviates from 1 the poorer is the solution.

Vertex cover is one of the graph related problem where the objective is to extract a set of vertices of a graph that covers all the edges of the graph. Minimal vertex cover is similar but here another objective is to optimize the solution such that the total vertices in the vertex cover set remain as minimal as possible.

In 1972, Richard Karp showed that finding the solution of minimal vertex cover in a graph is an NP-complete problem[3]. Thus, it is obvious that we can’t get optimal solution to MVC till it is proved that P=NP. Due to the existence of wide range of real life problems that can be formulated as MVC, various approximation and heuristics techniques have been developed and deployed by researchers. Vertex cover remains NP-complete even in cubic graphs[4] and even in planar graphs of degree at most 3[5]. Li et al argued that current heuristic algorithms of MVC only consider vertex features in isolation in order to decide whether a vertex is in or not in the solution set[6].

MVC cannot be approximated within a factor of 1.36, unless P=NP[7]. Numerous techniques and approximation algorithms have been presented in literature like 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. approximation solutions are poor. Some have a factor of 2-approximation while some are Δ-approximation where Δ is variable. Generally, 2-approximation algorithms are considered acceptable.

Literature Review

Richard Karp showed that Minimum vertex cover is NP-Complete[3]. It is widely believed that finding optimal solution to these problems is impossible in polynomial time. Chavatal proposed a simplest approximation algorithm for MVC which select a vertex randomly to be in MVC set, all adjacent edges are deleted and the process continues till no edge remains[7]. This was not a good approach because selection of node for MVC needs quite intelligent guess not a random guess. Clarkson modified this random approach and random selection was changed with selection made on the basis of degree[8]. The vertex with maximum degree is selected for MVC which gives better results than random guess. This approach was named MDG8. Run time complexity of this approach presented is in O (E2) where 'E' is total number of edges in a graph[8]. Its worst case approximation ratio is 'Δ' which is maximum degree in the graph. Delbot and Laforect experimentally analysed these approaches and among those MDG gives max of 33% error on ERDOS RENYI graphs, 9% on trees, 44% on BHOSLIB, 32% on regular graphs and 70% on average worst case graphs[9].

The key point in solving graphs for MVC is that it happens most of the time when we a select a node with maximum degree, it compels us to select extra nodes for covering all edges of graphs and affects the final outcome. Another greedy approach was presented by Chavatal which select a node with minimum degree[10]. This was originally presented for approximation of MIS but as MIS is also NP-Complete so MIS and MVC are reducible to each other, means we can solve these both problems on a single algorithm and practically it is simple because nodes other than MVC are MIS nodes[11]. Its run time complexity is in O (E2)[10]. It is mentioned by Halldarson and Radhakrishnan that GIC can find optimal solution in trees and therefore in paths[12].

List left was devised an approach which works on sorting all nodes in a list and then processes it from left to right and showed that its worst case approximation ratio is $\sqrt{\Delta/2}+3/2$ and minimum approximation ratio of $\sqrt{(\Delta/2)}$[13]. Experimental results presented by Delbot and Laforest shows that List left can't provide better or even same results compared to other algorithms implemented for analysis[9].

Delbot and Laforest Presented the same approach named List right with change in order of processing list, they process list from right to left[14]. Its maximum error percentage never exceeds 55 and provides better results than list left. Balaji et al devised a new approach with new data structure named support of a vertex[15]. All decisions regarding vertices are made on the basis of this value. Support of vertex that they proposed is the sum of degrees of all vertices adjacent to a vertex. They have tested their approach on large number of benchmarks and are optimal in most of the cases and its rum time complexity is O (EV^2). Li et al employed greedy approach in a different way names share of a vertex, where share of vertex is the total number of vertices it shares[6]. MVC node selection is made on the basis of this value but this approach not seems to be efficient on large graphs because of their complex data structure and calculations. A new clever intelligent greedy approach is presented by Gajurel and Bielefeld named NOVAC-1[16]. This approach works on a clever concept raised from the keen observation and analysis of relationship among vertices. The vertices attached to minimum degree nodes are candidate of MVC with high probability and they deployed this concept. Result shows that it provide optimal results on 35% of benchmark graphs tested and approximation ratio never exceeds 1.077 with an average approximation ratio of 1.008[16].

Proposed Algorithm

In the proposed algorithm we employed vertex support value which is data structure presented by S. Balajiet al[15]. According to them support of a vertex is the sum of degrees of all vertices adjacent to it or simply S (V) $=\sum_{u \in N}(v) d(u)$. The way vertices are connected to each other play an important role in taking decisions regarding any problem, so these relationships if modeled in efficient way can lead to optimal solutions in most of the graphs. The support value of a vertex is also such technique which not only accounts a single vertex but all vertices adjacent to it. The VSA algorithm works in same manner as MDG works, MDG select a vertex with maximum degree; VSA does the same but only selecting vertex with maximum support value. The selection of vertex can be critical because it effects future decisions. The more you filter vertices for decisions the more the result will be optimal. Our proposed algorithm also makes use of support value of a vertex but here we tried to make the decision as much intelligent and efficient as possible by taking account of all vertices adjacent to it, we can also refer it as random sub graph. Not only the vertex with maximum support value is candidate of MVC but vertex attached to minimum degree vertices is also important for it.

Working of Proposed Algorithm

Working of proposed algorithm is divided into three main steps; first step is calculation of values for analysis. In this step we first calculate degree of each vertex followed by calculation of support value of each vertex. At completion of this step a data structure is ready to be manipulated for decisions regarding vertices. Second step is filtering step in which vertices are filtered so that selection can be made on the basis of sub graphs. Filtration process is carried out by creation of two set of nodes, 'min_support' as first set contains all nodes which have minimum support value and 'adj_nodes' second set contains all nodes adjacent to nodes in the first set. This filtration helps in minimizing the after effects of decisions for the whole graph.

Third step is selection step in which a vertex is selected as candidate node for MVC, selection is made from 'adj_nodes' and vertex with minimum support value is selected as candidate node. All edges adjacent to candidate node are deleted and the process continues till no edge remains. Computation model designed for MVSA is very simple in both understanding and implementation. The pseudo code for the proposed MVSA algorithm is given below.

Algorithm_ MVSA (Graph G)
{
MVC [],
D [],
S [],
min_support []
While (Edges ≠ ∅)
{
For each v ∈ V calculate degree of v
// calculate degree for each node in the graph.
For each v ∈ V
Calculate support value for each v that is S (V) $=\sum_{u\in N}(v)d(u)$.
Find out all nodes with minimum support value and add it to 'min_support'
Find out neighbors to all nodes in 'min_support'.
Find node with minimum support value in neighbors of 'min_support'.
Add it to MVC, Drop all its edges.
}
}

There is no extra complexity involved in computation and decisions are made straight forward. Run time complexity of the proposed algorithm is in O (EV log v).

Empirical Results

To analyze efficiency of MVSA extensive experiments to solve a large class of benchmarks are carried out and not only this but MVSA is also compared with other well-known algorithms present in literature. All implementation tasks are carried out in Mat lab on core i3 system running windows 8. Extensive experimental results shows that worst case approximation ratio of MVSA is '1.064' with an average approximation ratio of '1.008'. Table1 outlines these experimental results, first column contains benchmarks tested, second is total number of vertices, third is for optimal solution, fourth is for results obtained through MVSA, fifth is for results obtained through MDG, sixth is for results of NOVAC-1, seventh main column is for approximation ratios one each for MVSA, MDG and NOVAC-1

Results shown in table1 are analyzed thoroughly for efficiency check and it witness that MVSA, VSA, MDG and NOVAC-I give worst approximation ratio of 1.064, 1.583, 1.107 and 1.049 respectively. This analysis shows that results of MVSA are good as compared VSA and MDG. Table1 was also analyzed for extraction of average approximation ratio and it is drawn that MVSA is capable of solving it with in 1.008 on average. VSA, MDG and NOVAC-1 give average approximation ratio of 1.0608, 1.012 and 1.006 respectively. The way at which vertices are selected is changed and we got a measurable difference in both worst and average case. Figure1 visualizes the comparison of these algorithms and diversion from straight line shows the distance between optimal and gained solutions. Diversion in VSA is very high in certain graphs; some of these results are given by S.balaji[15] while remaining results are obtained by self-implementing the procedure of vertex support algorithm. Among all MVSA perform best on average which can be observed from figure1.

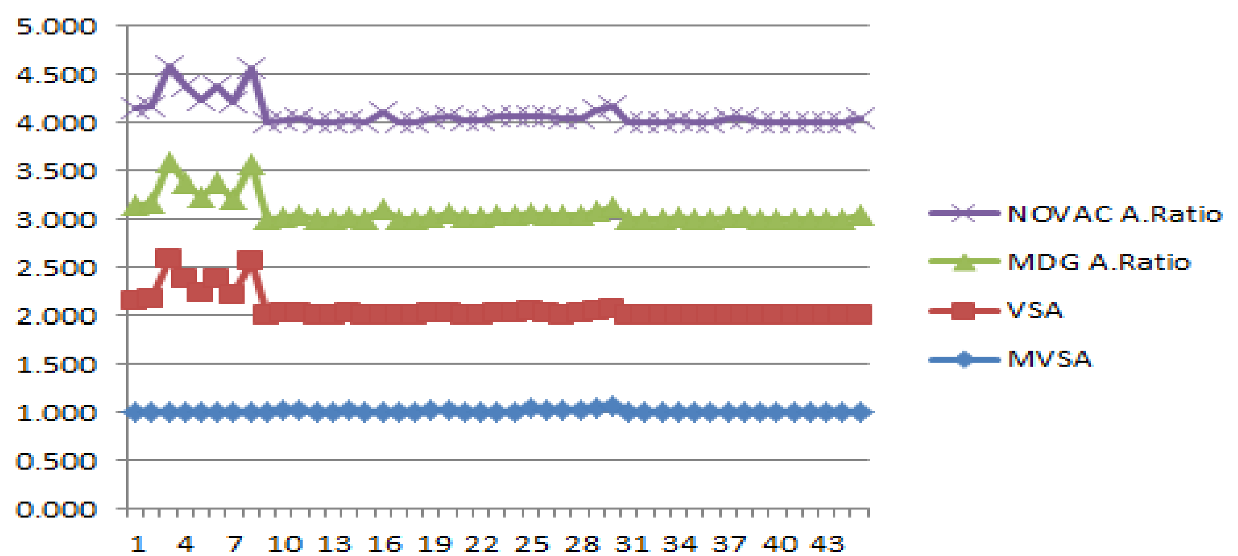

Figure-1
Approximation ratio comparison of NOVAC, MDG and MVSA

Table-1
Experimental results of MVSA, MDG and NOVAC-1 against benchmark graphs.

Benchmarks	***Total Vertices***	***Optimal MVC***	***MVSA***	***VSA***	***MDG***	***NOVAC-1***	***Approximation Ratios***			
							MVSA	***VSA***	***MDG***	***NOVAC-1***
graph50_6	50	38	38	44	38	38	1.000	1.158	1.000	1.000
graph50_10	50	35	35	41	35	35	1.000	1.171	1.000	1.000
graph100_1	100	60	60	95	60	60	1.000	1.583	1.000	1.000
graph100_10	100	70	70	96	70	70	1.000	1.371	1.000	1.000
graph200_5	200	150	150	184	150	150	1.000	1.227	1.000	1.000
graph500_1	500	350	350	485	350	350	1.000	1.386	1.000	1.000
graph500_2	500	400	400	484	400	400	1.000	1.210	1.000	1.000
graph500_5	500	290	290	454	290	290	1.000	1.566	1.000	1.000
phat300_1	300	292	294	292	293	293	1.007	1.000	1.003	1.003
phat300_2	300	275	279	275	278	275	1.015	1.000	1.011	1.000
phat300_3	300	264	272	264	269	266	1.030	1.000	1.019	1.008
phat700_1	700	689	692	689	693	692	1.004	1.000	1.006	1.004
phat700_2	700	656	660	656	660	657	1.006	1.000	1.006	1.002
phat700_3	700	638	649	638	642	641	1.017	1.000	1.006	1.005
jhonson8_2_4	28	24	24	24	24	24	1.000	1.000	1.000	1.000
jhonson8_4_4	70	56	56	56	62	56	1.000	1.000	1.107	1.000
jhonson16_2_4	120	112	112	112	112	112	1.000	1.000	1.000	1.000
jhonson32_2_4	496	480	480	480	480	480	1.000	1.000	1.000	1.000
sanr200-0.7	200	182	186	182	184	185	1.022	1.000	1.011	1.016
sanr200-0.9	200	158	163	158	164	159	1.032	1.000	1.038	1.006
sanr400_0.5	400	387	389	387	392	388	1.005	1.000	1.013	1.003
sanr400_0.7	400	379	381	379	384	381	1.005	1.000	1.013	1.005
fbr_30_15_5	450	420	424	429	429	424	1.010	1.021	1.021	1.010
fbr_35_17_2	595	560	565	573	570	565	1.009	1.023	1.018	1.009
c 125	125	91	95	91	93	92	1.044	1.000	1.022	1.011
c 250	250	206	211	206	211	211	1.024	1.000	1.024	1.024
c500.9	500	≤443	449	443	453	449	1.014	1.000	1.023	1.014
broc200_2	200	188	191	188	192	190	1.016	1.000	1.021	1.011
broc200_4	200	183	193	183	188	192	1.055	1.000	1.027	1.049
gen200_p0.9_44	200	156	166	156	165	163	1.064	1.000	1.058	1.045
Hamming6_2	64	32	32	32	32	32	1.000	1.000	1.000	1.000
Hamming6_4	64	60	60	60	60	60	1.000	1.000	1.000	1.000
Hamming8_2	256	128	128	128	128	128	1.000	1.000	1.000	1.000
Hamming8_4	256	240	240	240	248	240	1.000	1.000	1.033	1.000
Hamming10_2	1024	512	512	512	512	512	1.000	1.000	1.000	1.000
dsjc500	500	487	489	487	491	488	1.004	1.000	1.008	1.002
keller4	171	160	160	160	164	164	1.000	1.000	1.025	1.025
keller5	776	749	754	749	764	761	1.007	1.000	1.020	1.016
cfat200_1	200	188	188	188	188	188	1.000	1.000	1.000	1.000
cfat200_2	200	176	176	176	176	176	1.000	1.000	1.000	1.000
cfat200_5	200	142	142	144	142	142	1.000	1.014	1.000	1.000
cfat500_1	500	486	486	486	486	486	1.000	1.000	1.000	1.000
cfat500_2	500	474	474	474	474	474	1.000	1.000	1.000	1.000
cfat500_5	500	436	436	436	436	436	1.000	1.000	1.000	1.000
mann_a27	378	252	253	253	261	253	1.004	1.004	1.036	1.004

Conclusion

Modified form of vertex support algorithm is presented which is able to solve benchmark graphs better than the original VSA on average. The proposed algorithm is applied against all best available benchmarks in order to prove its efficiency on as much solid basis and possible. In future we will work on making it more efficient with as much simple design as possible. We will also extend dimension of our work to weighted graphs where minimum weighted vertex cover is the sum of weight of vertex cover nodes and aim is to make it as much minimum as possible.

References

1. Chen J. et al, Linear FPT reductions and computational lower bound, *Proc.* 36th ACM symposium on Theory of computing, *New York,* 212-221, **(2004)**

2. Garey M. and Johnson D., *Computers and intractability, New York: Freeman,* **(1979)**

3. Karp R., Reducibility among combinatorial problems, *New York: Plenum Press,* **(1972)**

4. Demaine E. et al, Sub exponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs, *Journal of the ACM (JACM)*, **52(6)**, 866-893, **(2005)**.

5. Dinur I and Safra S., on the hardness of approximating minimum vertex cover, *Annals of Mathematics,* **(162)**, 439-485 **(2005)**

6. Li S., et al, An Approximation Algorithm for Minimum Vertex Cover on General Graphs, *in Proc. 3rd International Symposium: Electronic Commerce and Security Workshops (ISECS '10),* China: Academy, 249-252, **(2010)**

7. Chvatal V., A Greedy Heuristic for the Set-Covering Problem, *Mathematics of Operations Research*, **(4)**, 233-235, **(1979)**

8. Clarkson K., A modification to the greedy algorithm for vertex cover problem, *IPL*, **(16)**, 23-25, **(1983)**

9. Delbot F. and Laforest C., Analytical and experimental comparison of six algorithms for the vertex cover problem, *ACMJ. Experimental Algorithmics,* **(15),** 1-4 **(2010)**

10. Chvatal V., A Greedy Heuristic for the Set-Covering Problem, *Mathematics of Operations Research*, **(4)**, 233-235, **(1979)**

11. Cormen T.H., Lieserson C.E, Rivest R.L. and Stein C., Introduction to Algorithms, 3rd Ed, MIT Press England, **(2009)**

12. Halldorsson M. and Radhakrishnan J., Greed is good: Approximating independent sets in sparse and bounded-degree graphs, In Proc. 26th Annual ACM Symposium on Theory of Computing, *New York: ACM,*439–448, **(1994)**

13. Avis D. and Imamura T., A List Heuristic for Vertex Cover, *Operations research letters*, **35(2),** 201-204, **(2007)**

14. Delbot F. and Laforest C., A better list heuristic for vertex cover*, Information Processing Letters,* **(107)**, 125-127,**(2008)**

15. Balaji S., Swaminathan V. and Kannan K., Optimization of Un-weighted Minimum Vertex Cover, *World Academy of Science, Engineering and technology,* **(67)**, 508-513 **(2010)**

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