Computer Technology and Application 5 (2014) 83-90

A Heuristic Approach to Fast NOVCA (Near Optimal Vertex Cover Algorithm)

Sanjaya Gajurel and Roger Bielefeld

ITS, Advanced Research ComputingCWRU, Cleveland 44106, USA

Abstract: This paper describes an extremely fast polynomial time algorithm, the NOVCA （Near Optimal Vertex Cover Algorithm） that produces an optimal or near optimal vertex cover for any known undirected graph G (V, E). NOVCA is based on the idea of (1) including the vertex having maximum degree in the vertex cover and (2) rendering the degree of a vertex to zero by including all its adjacent vertices. The three versions of algorithm, NOVCA-I, NOVCA-II, and NOVCA-random, have been developed. The results identifying bounds on the size of the minimum vertex cover as well as polynomial complexity of algorithm are given with experimental verification. Future research efforts will be directed at tuning the algorithm and providing proof for better approximation ratio with NOVCA compared to any available vertex cover algorithms.

Key words: Vertex cover problem, combinatorial problem, NP-complete problem, approximation algorithm, optimization, algorithms.

1. Introduction

The VC （vertex cover） of a graph G(V, E) with vertex set V and edge set E is a subset of vertices C of V (C ⊆ V) such that every edge of G has at least one endpoint in C. In 1972, Richard [1] showed that identification of minimal VC in a graph is an NP-complete problem.

Various algorithmic approaches have been used to tackle NP-complete problems. The vertex cover problem has been actively studied because of its important research and application implications. Polynomial-time approximation and heuristic algorithms for VC have been developed but none of them guarantee optimality. By using the definition of approximation ratio, VC has an approximation ratio of ρ (n) for any input of size n. The solution C produced by approximation algorithm is within the factor of ρ(n) of the solution c* of an optimal algorithm， i.e., C*/C ≤ ρ (n). Also, the approximation algorithm has approximation ratio of 2 – ε, where 0 < ε < 1. A 2-approximation [2] algorithm has been trivially obtained and similar approximation algorithms have been developed [3, 4] with an approximation of (2 – (ln (ln n)/2ln n)), where n is the number of vertices. Halperin [5] achieved an approximation factor of (2 – (1 – o (1))(2ln (ln Δ)/ ln Δ)) with maximum degree at most Δ. Karakostas [6] attained an approximation factor of (2 – θ(1/(log n)1/2))), the best approximation yet, by using the semidefinite programming relaxation of VC. EA (evolutionary algorithms) that are randomized search heuristics have also been used for solving combinatorial optimization problems including VC [7, 8].

Vertex cover problems have been solved in O (1.2738k + kn) time [9] by using a bounded search technique where a function of a parameter restricts the search space. Abu-Khazm et al. [10] have identified crown structure to reduce the size of both n and k. It has been known that when relevant parameters are fixed, NP-complete problems can be solved in polynomial time. In both Ref. [10] and Ref. [11], n is the input size and k is the positive integer parameter. Though not guaranteed to find a minimum vertex

Corresponding author: Sanjaya Gajurel, Ph.D., research fields: HPC, MANET, swarm algorithm, optimization. E-mail: sxg125@case.edu.

cover, an approximation of 3/2 for almost every single graph was obtained in Ref. [11]. According to Ref. [12], it is NP-hard to get $\varepsilon < 1.3606$.

The paper is organized as follows: the NOVCA algorithm is described in Section 2; Section 3 provides experimental results; Section 4 is the conclusion.

2. Near Optimal Vertex Cover Algorithm

NOVCA is motivated by the fact that vertex cover candidates are those that are adjacent to minimum degree vertex so that its degree will be forcibly rendered to zero without choosing it. This fact has been reinforced during tie when the vertex with neighbors having maximum degrees is preferred over other minimum vertices. Without any optimization effort, the complexity of NOVCA is O (E ($V + \log^2 V$)); with V = n, the complexity becomes O (n^2 ($n + \log^2 n$)) which is polynomial. Network Bench Node Degree algorithm [13] has been applied to determine the degree of each node. Then, the sum of the degree of adjacent nodes for each node is calculated. Both these values are included as data structures in a node - deg[v]/adj_deg_sum[v] as showed in pseudo-code of NOVCA algorithm as follows.

Input: *V is the set of vertices of G, E is the set of edges of G, deg[V] is an integer array indexed by V for a set of vertices V, sum_adj_deg[V] is an integer array indexed by V for a set of vertices V, Qsum_adj_deg is the set of vertices having min deg[V].*

Output: *VC is the set of vertices comprising a vertex cover.*

Functions: *Degree(v) is the degree of the vertex v ϵ V, Adj(v) gives the set of vertices that are adjacent to v ϵ V, GetMinVertex() identifies the next adjacent vertices to include in the cover, Heap_MIN(deg) returns the value of min. deg[V], HEAP_MAX(Qsum_adj_deg) returns the vertex having max Qsum_adj_deg*

```
for each v ϵ V {
deg[v] = Degree(v)
}
for each v ϵ V {
sum_adj_deg[v] =Σ v'εAdj(v)deg[v']
}
E' = E
VC = ф
while (E'≠ ф){
vc = GetMinVertex(deg, sum_adj_deg)
VC = VC + { Adj(vc) }
for each v ϵ Adj(Adj(vc)){ //for NOVCA-I
//for each v ϵ Adj(vc){ //for NOVCA-II
E' = E – { (adj(vc), v) }
deg[v] = deg[v] – 1
}
V = V – { Adj(vc) } //for NOVCA-I
//V = V – { vc } //for NOVCA-II
for each v ϵ V{
If (Adj(v) == ф) continue
sum_adj_deg[v] = Σ v'εAdj(v)deg[v']
}
} //end while
/// Magic Function GetMinVertex() Declarations ///
Vertex GetMinVertex(deg, sum_adj_deg){
Qsum_adj_deg= ф
vmin_deg = HEAP_MIN(deg) //for NOVCA-I
//vmax_deg= HEAP_MAX(deg)//for NOVCA-II
for each v ϵ V{
If (deg[v] == vmin_deg) //for NOVCA-I
//If (deg[v] == vmax_deg) //for NOVCA-II
Qsum_adj_deg = Qsum_adj_deg + {v}
}
returnHeap_MAX(Qsum_adj_deg)//for NOVCA-I
//return Heap_MIN(Qsum_adj_deg) //for NOVCA-II
}
```

Initially, vertex cover set VC is empty. NOVCA-I [14] constructs the vertex cover by repeatedly adding, at each step, all vertices adjacent to the vertex of minimal degree. In the case of a tie, it chooses the one having the maximum sum of degrees of its neighbors. NOVCA-II [15], on the other hand, builds vertex

cover by including vertices in descending order of degree and in the case of a draw; it selects the vertex having the minimum sum of degrees of its neighbors. The magic function GetMinVertex breaks a tie in selecting the best candidate vertex in a vertex cover. The implementation forcibly renders the degree of low degree vertices to zero without choosing them.

The version, NOVCA-random, selects NOVCA-I and NOVCA-II based on the random value, i.e., NOVCA-I and NOVCA-II are chosen with the probability of 85% and 15% respectively during each RUN as NOVCA has showed better performance in most of the instances.

3. Experiment Works and Results

Experiments to corroborate the theoretical results have been conducted on the CWRU HPC (high performance computing) resource using compute nodes with 3.0 GHz Intel Xeon processors running Red Hat Enterprise Linux 4 and using the gcc 3.4.6 compiler. We have selected complete graph as a test graph to determine time complexity of NOVCA for two reasons:

- Optimal vertex cover is known, $n - 1$, where n is the number of vertices;
- requires exhaustive search; there is an edge from each vertex to all other vertices.

The shell script "graph_gen.sh" generates a complete graph of size n entered as input. This graph is then fed to executable "vc" (C++ program compiled with g++ compiler) to get vertex cover for that particular graph.

```
#PBS -l walltime=01:00:00
#PBS -l nodes=1:ppn=1
#PBS -N graph1000
#PBS -j oe
cd $PBS_O_WORKDIR
/usr/local/bin/pbsdcp -s vc graph_gen.sh $TMPDIR
cd $TMPDIR
sh graph_gen.sh 1000
cpgen_graph graph1000
time ./vc graph1000
/usr/local/bin/pbsdcp -g '*' $PBS_O_WORKDIR
cd $PBS_O_WORKDIR
```

We have recorded the computation time for different sizes of the graphs to elucidate the polynomial complexity of NOVCA algorithm through MATLAB's polyfit(x,y,n) command as showed in Figs. 1and 2.

NOVCA has approximation ratio smaller than 1.3606 for all available bench mark (Table 1, Table 2) and Table 3 [16]; all instances, however, are not showed here. For some instances like c-fat, Johnson, and random graphs NOVCA provides optimal cover. Noticeably, the execution time of NOVCA for any instance is remarkable. NOVCA has been found to perform very well compared to other available algorithms. For the instances where it provides near optimal solutions, it outperforms other algorithms in

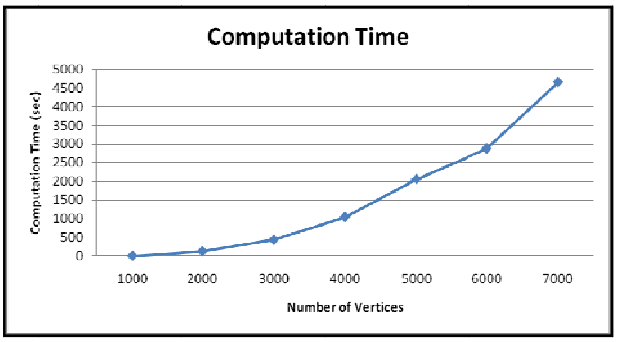

Fig. 1 Computational time of NOVCA for different sizes of complete graphs.

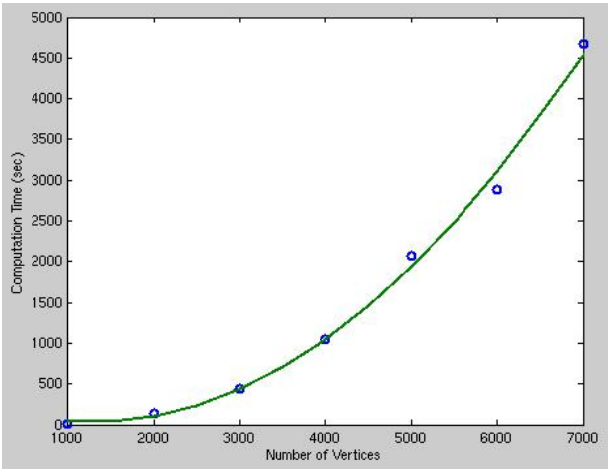

Fig. 2 MATLAB plot using polyfit with $n = 2$.

Table 1 DIMACS and BHOSLIB benchmarks.

Instances	$\|V\|$	$\|C^*\|$
frb59-26-1	1,534	1,475
frb59-26-2	1,534	1,475
frb100-40	4,000	3,900
broc200_1	200	179
broc800_4	800	774
C2000.9	2,000	1,922
c-fat200-5	200	142
c-fat500-10	500	374
gen200_p0.9_44	200	156
hamming10-2	1,024	512
hamming10-4	1,024	984
johnson16-2-4	120	112
johnson32-2-4	496	480
keller4	171	160
keller5	776	749
MANN_a27	378	252
MANN_a81	3,321	2,221
p_hat500-1	500	491
p_hat1500-3	1,500	1,406
san200_0.7_1	200	170
san1000	1,000	985
sanr200_0.7	200	183
sanr400_0.7	400	379
graph50-10	50	35
graph50-10	50	35
graph100-10	100	70
graph200-05	200	150
graph250-05	250	200
graph500-05	500	290

terms of execution time. We have compared NOVCA with COVER [17]. COVER is a stochastic local search algorithm for k-vertex cover. It constructs the initial candidate solution *C* greedily. When the several vertices satisfy the criterion for inclusion in *C*, COVER selects one of them randomly with uniform probabilities. The COVER algorithm terminates when either the vertex cover is found or max number of steps (MAX_ITERATIONS) has been reached. NOVCA-I and NOVCA-II, on the other hand, do not have any randomness element and terminate when there are no more vertices in *V*. So, they have only one run unlike average execution time calculated using random seeds in different runs in COVER. NOVCA-random has randomness only in selection of two algorithmic approaches.

Though COVER is found to obtain better vertex cover in most of the instances of the benchmarks, NOVCA is very simple and it outperforms COVER in execution time. In case of the graph instance, MANN_a81, where both NOVCA and COVER return the same value 2,225, NOVCA is 20 times faster. Though NOVCA-I outperforms NOVCA-II in terms of approximation ratio in almost all instances except keller, p-hat, and sanr, NOVCA-II has better execution

Table 2 Performance comparison between NOVCA-I and COVER on DIMACS and BHOSLIB benchmarks $|V|$: number of vertices; $|C^*|$: optimal cover; NOVCA $|C|$: cover returned by NOVCA; COVER $|C|$avg. cover returned by COVER; NOVCA Time (s): execution time for NOVCA; COVER Time_{avg}: average execution time for COVER; no data available for the instance frb100-40 in COVER.

NOVCA-I $\|C\|$	NOVCA-I $\|C\|/\|C^*\|$	NOVCA-I Time (s)	COVER $\|C\|_{avg}$	COVER Time_{avg} (sec)
1,485	1.007	80.258	1,477	1,8611.3
1,484	1.006	79.297	1,478	1,8589.5
3,917	1.004	2013.667	-	-
181	1.011	0.115	179	768.2
782	1.010	10.832	775	4,051.2
1,932	1.005	207.060	1,922	21,489.7
142	1	0.092	142	1,549.1
374	1	2.117	374	4,401.2
163	1.045	0.092	156	1,543.6
512	1	10.297	512	2,412.2
988	1.004	21.505	986	3,457.6
112	1	0.076	112	297.9
480	1	2.273	480	2,351.9
164	1.025	0.007	160	985.7
761	1.016	9.125	749	2,364.9
253	1.004	0.493	252	756.3
2,225	1.002	773.963	2,225	15,672.1
492	1.002	2.683	491	1,810.2
1,414	1.006	74.991	1,406	1,298.9
183	1.077	0.117	170	713.7
991	1.006	22.901	989	4,972.8
185	1.011	0.857	183	788.2
382	1.008	1.030	380	2112.5
35	1	0.006	35	124.5
70	1	0.034	70	205.3
150	1	0.114	150	854.1
200	1	0.300	200	988.5
290	1	1.604	290	22,555.2
1,485	1.007	80.258	1,477	18,611.3

Table 3 Performance comparison between NOVCA-II and COVER on DIMACS and BHOSLIB benchmarks $|V|$: number of vertices; $|C^*|$: optimal cover; NOVCA $|C|$: cover returned by NOVCA; COVER $|C|_{avg}$: cover returned by COVER; NOVCA Time (s): execution time for NOVCA; COVER $Time_{avg}$: average execution time for COVER; no data available for the instance frb100-40 in COVER.

NOVCA-II $\|C\|$	NOVCA-II $\|C\|/\|C^*\|$	NOVCA-II Time (s)	COVER $\|C\|_{avg}$	COVER $Time_{avg}$ (s)
1,494	1.014	34.770	1,477	18,611.3
1,496	1.014	35.686	1,478	18,589.5
3,944	1.011	885.860	-	-
182	1.017	1.316	179	768.2
786	1.016	6.162	775	4,051.2
1,942	1.010	88.604	1,922	21,489.7
142	1	1.238	142	1,549.1
374	1	1.514	374	4,401.2
170	1.090	1.514	156	1,543.6
512	1	5.584	512	2,412.2
992	1.008	10.350	986	3,457.6
112	1	1.248	112	297.9
480	1	2.245	480	2,351.9
162	1.013	1.500	160	985.7
761	1.016	5.115	749	2,364.9
261	1.036	1.641	252	756.3
2,241	1.009	297.236	2,225	15,672.1
492	1.002	2.595	491	1,810.2
1,412	1.004	34.535	1,406	1298.9
185	1.088	1.535	170	713.7
992	1.007	11.657	989	4,972.8
184	1.005	1.351	183	788.2
384	1.013	1.947	380	2,112.5
35	1	1.667	35	124.5
70	1	1.552	70	205.3
150	1	1.523	150	854.1
200	1	1.653	200	988.5
290	1	2.366	290	22,555.2
1,494	1.014	34.770	1,477	18,611.3

time than NOVCA-I. NOVCA-random always returns better cover than both NOVCA-I and NOVCA-II. For the challenge instances of frb100-40 [16], NOVCA-I is off by just 17 vertices (NOVCA returns 3,917 vertices whereas the optimal vertex cover is 3,900), but the execution time is just remarkable, only 2,013.667 s. The challenge is stated as "Based on theoretical analysis and experimental results of smaller instances, I conjecture that in the next 20 years or more (from 2005), these two benchmarks cannot be solved on a PC (or alike) in a reasonable time (e.g., 1 day) [16]." The graphs for number of vertices returned and the execution times, as showed in Figs. 3 and 4 respectively, portray that NOVCA, though comparable to COVER in terms of number of vertices returned, is significantly faster than COVER.

Table 4 Performance of NOVCA-random on DIMACS and BHOSLIB benchmarks $|V|$: number of vertices; $|C^*|$: optimal cover; NOVCA $|C|$: cover returned by NOVCA-I and NOVCA-II (one RUN); $|C|_{min}$: minimum number of vertices returned by NOVCA-random out of 1,000 RUNS.

NOVCA-I $\|C\|$	NOVCA-II $\|C\|$	NOVCA-random $\|C\|_{avg}$
1,485	1,494	1,479
1,484	1,496	1,479
3,917	3,944	3,904
181	182	180
782	786	780
1,932	1,942	1,932
142	142	142
374	374	374
163	170	162
512	512	512
988	992	984
112	112	112
480	480	480
164	162	162
761	761	754
253	261	253
2,225	2,241	2,221
492	492	491
1,414	1,412	1,408
183	185	183
991	992	990
185	184	183
382	384	380
35	35	35
70	70	70
150	150	150
200	200	200
290	290	290
1,485	1,494	1,479

Table 4 compares all the versions of NOVCA which are then represented in bar diagrams in Fig. 5 to visualize the better performance of NOVCA-random compared to the earlier versions, NOVCA-I and NOVCA-II, based on the minimum number of vertices obtained from 1,000 runs. For the instance

frb100-40, the minimum vertex cover returned by NOVCA-random ($|C|_{min}$ = 3,904) is considerably smaller than the covers returned by NOVCA-I (|C| = 3,917) and NOVCA-II ($|C|$ = 3,944). We have also carried out comparisons of NOVCA against two other heuristic MVC (minimum vertex cover) algorithms, PLS [18] and EWCC [19], with similar results (not explicitly tabulated here).

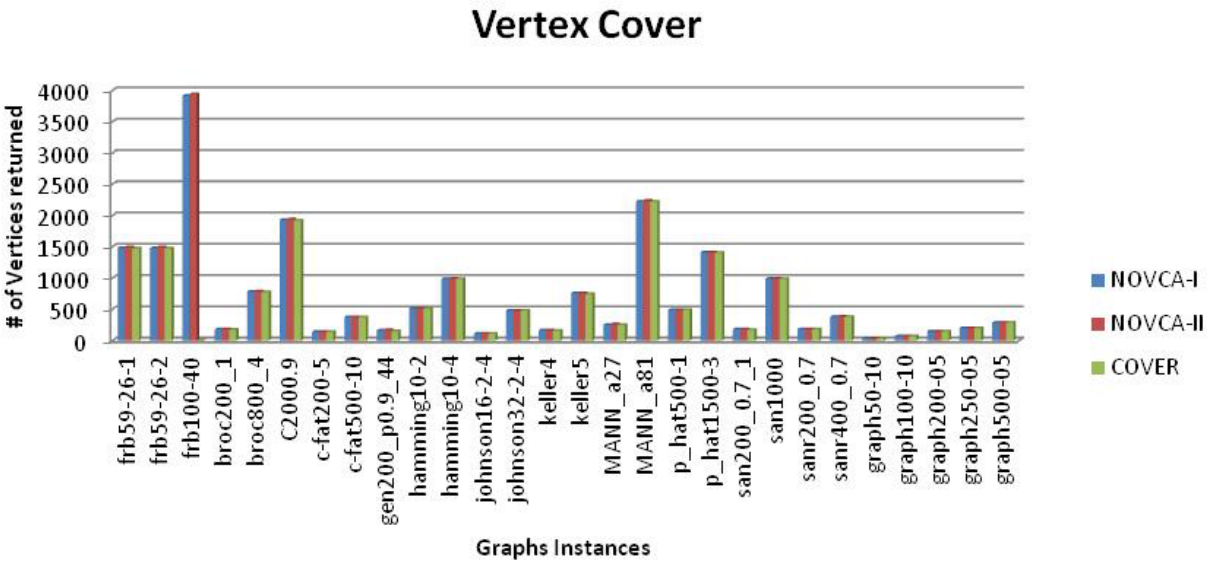

Fig. 3 Number of vertices returned by NOVCA-I, NOVCA-II, and COVER; no results from COVER for the instance frb100-40.

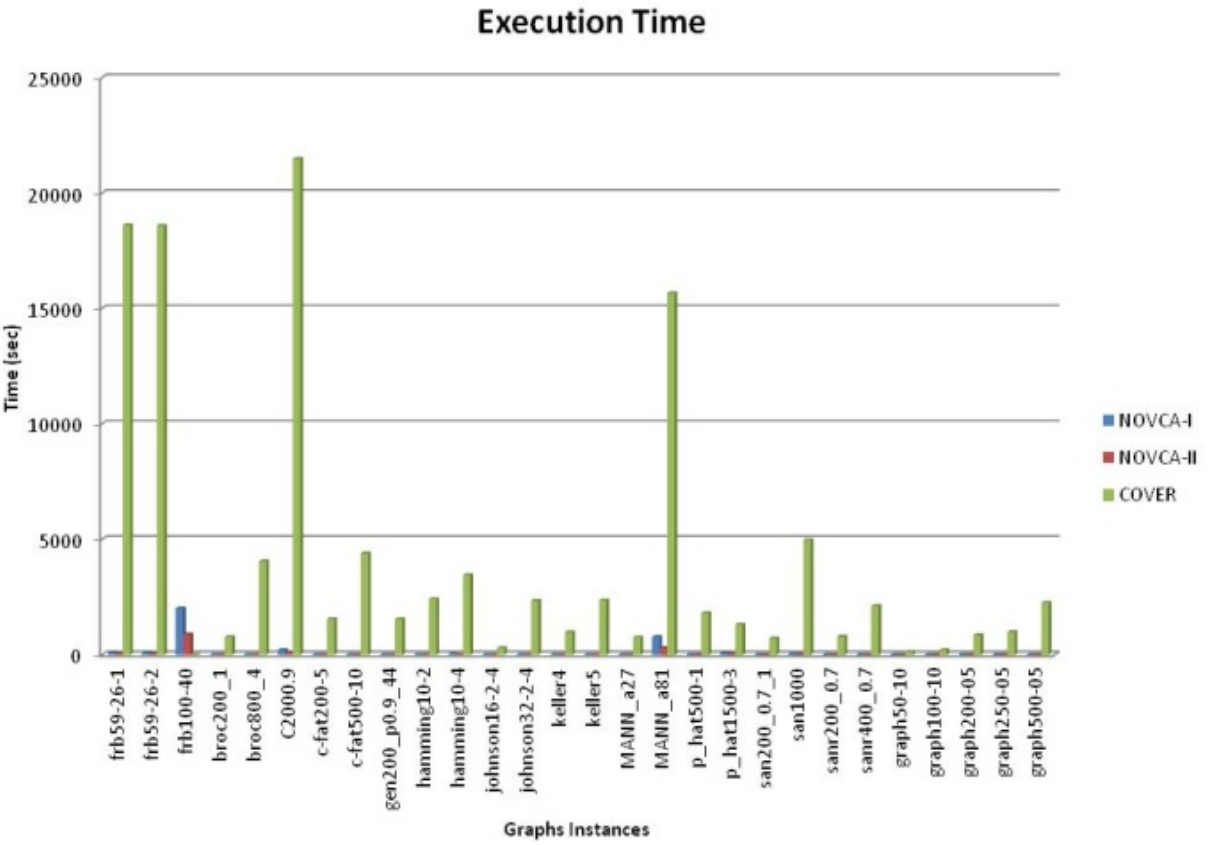

Fig. 4 Execution time for NOVCA-I, NOVCA-II, and COVER; no results from COVER for the instance frb100-40.

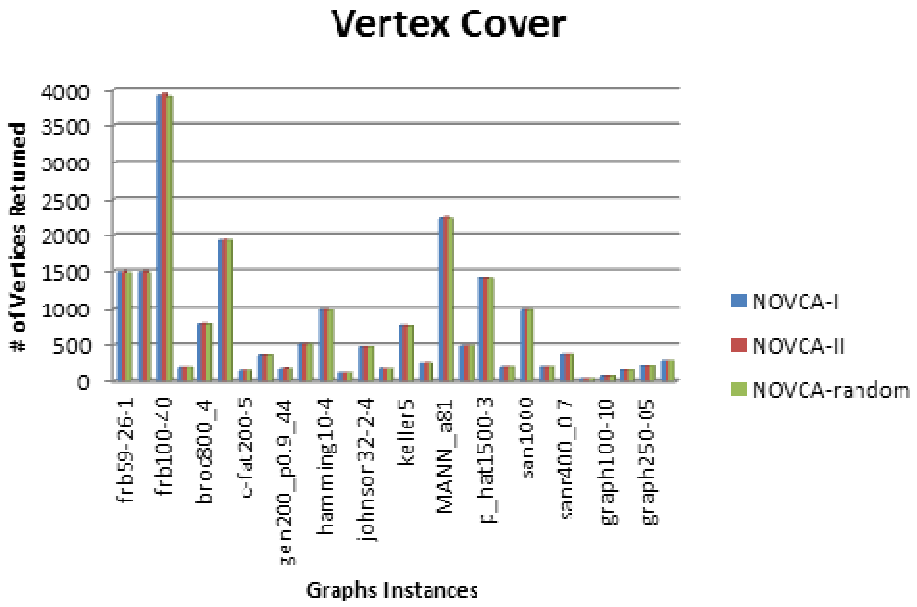

Fig. 5 Number of vertices returned by NOVCA-I, NOVCA-II, and NOVCA-random.

4. Conclusions and Future Work

All the versions of NOVCA algorithms, NOVCA-I, NOVCA-II, and NOVCA-random, provide optimal or near optimal vertex cover for known benchmark graphs. The experimental results depict that NOVCA is extremely fast compared to other available state-of-the-art MVC algorithms including COVER, PLS, and EWCC.

Future research will be focused in two areas: deriving a mathematical statement regarding the closeness of the approximation ratio to 1, and investigating approaches to parallelizing the NOVCA algorithm.

Acknowledgments

This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University.

References

[1] Karp, R., 1972. "Reducibility among Combinatorial Problems." In *Proceedings of a Symposium on the Complexity of Computer Computations,* 85-103.

[2] Cormen, T., Leiserson, C., and Rivest, R. 2001. *Introduction to Algorithms.* United States :The MIT Press.

[3] Bar-Yehuda, R., and Even, S. 1985. "A Local-Ratio Theorem for Approximating the Weighted Vertex Cover Problem." *Annals of Discrete Mathematics* 25: 27-45.

[4] Monien, B., and Speckenmeyer, E. 1985. "Ramsey Numbers and an Approximation Algorithm for the Vertex Cover Problem." *ActaInformatica* 22: 115-23.

[5] Halperin, E. 2000. 2002. "Improved Approximation Algorithms for the Vertex Cover Problem in Graphs and Hypergraphs." *SIAM J. on Computing* 31 (5): 1608-23.

[6] Karakostas, G. 2005. "A Better Approximation Ratio for the Vertex Cover Problem." In *Proceedings of the ICALP,* 1043-50.

[7] Rudolph, G. 1998. "Finite Markov Chain Results in Evolutionary Computation." *A tour d'horizon, Fundamenta Informaticae* 35 (1-4): 67-89.

[8] Oliveto, P., He, J., and Yao, X. 2007. "Evolutionary Algorithms and the Vertex Cover Problem." In *Proceedings of the IEEE Congress.*

[9] Chen, J., Kanj, I., and Xia, G. 2005. *Simplicity Is Beauty: Improved Upper Bounds for Vertex Cover.* Technical report, Texas A&M University.

[10] Abu-Khazm, F., Fellows, M., Langston, M., and Suters, W. 2007. *Crown Structures for Vertex Cover Kernelization.* Vol. 41. Egypt:Theory Comput. Systems, 411-30.

[11] Asgeirsson, E., and Stein, C. 2007. "Vertex Cover Approximation on Random Graphs." *LNCS* 4525: 285-96.

[12] Dinur, I., and Safra, S. 2001. *The importance of being biased.* Technical Report TR01-104, ECCC.

[13] Network Bench Node Degree, 2006. "http://nwb.slis.indiana.edu/"

[14] Gajurel, S., and Bielefeld, R. 2012. "A Simple NOVCA: Near Optimal Vertex Cover Algorithm." *Procedia Computer Science* 9: 747-53.

[15] Gajurel, S., and Bielefeld, R. 2012. "A Fast near Optimal Vertex Cover Algorithm (NOVCA)." *IJEA* 3 (1): 9-18.

[16] Xu, K. 2012. "Vertex Cover Benchmark Instances (DIMACS & BHOSLIB)." IJEA (international journal of Eexperimental algorithms) 3 (1): 1-18. Accessed October 25, 2012. http://www.cs.hbg.psu.edu/benchmarks/vertex_cover.html.

[17] Richter, S., Helmert, M., and Gretton, C. 2007. "A Stochastic Local Search Approach to Vertex Cover." In *Proceedings of the 30th German Conference of Artificial Intelligence (KI),* 412-26.

[18] Cai, S., Su, K., and Sattar, A. 2011. "Local Search with Edge Weighting and Configuration Checking Heuristics for Minimum Vertex Cover." *Artifical Intelligence* 175: 1672-96.

[19] Pullan, W. 2006. "Phased Local Search for the Maximum Clique Problem." *Journal of Combinatorial Optimization* 12: 303-23.