New Heuristic Algorithm for Unweighted Minimum Vertex Cover

Onur Ugurlu
Ege University, Izmir, Turkey
onur__ugurlu@hotmail.com

***Abstract*— The minimum vertex cover (MVC) problem belongs to the class of NP-complete graph theoretical problems, which plays a central role in theoretical computer science and it has numerous real life applications. We are unlikely to find a polynomial-time algorithm for solving the vertex-cover problem exactly. In this paper, a fast heuristic algorithm, called Isolation Algorithm (IA), has been designed to find the minimum vertex cover of a graph. The IA has been tested on DIMACS benchmark graphs and BHOSLIB instances. The results of the computational experiments have shown that the IA can yield better solutions on small size graphs for solving the minimum vertex cover problem.**

Keywords*— *NP-complete problem; optimization; vertex cover; heuristic algorithms

I. INTRODUCTION

A vertex cover of an undirected graph G=(V,E)is a subset $V^* \subseteq V$ such that if $(u,v) \in E$, then $u \in V^*$ or $v \in V^*$(or both). That is, each vertex "covers" its incident edges, and a vertex cover for G is a set of vertices that covers all the edges in E. The size of a vertex cover is the number of vertices in it [1]. The problem of finding a minimum vertex cover is a classical optimization problem in computer science and is a typical example of an NP-complete optimization problem. Karp showed that the vertex cover problem is NP–complete in a landmark paper, in 1972 [2]. According to the NP-completeness theory [3], this problem cannot be solved in polynomial time unless P = NP. Minimum vertex cover has attracted researchers and practitioners not only because of the NP–completeness but also because of many difficult real-life problems which can be formulated as instances of the minimum vertex cover. Examples of such areas where the minimum vertex cover problem occurs in the real world applications are communications, particularly in wireless telecommunications, civil, electrical engineering, especially in multiple sequence alignments for computational biochemistry [4].

Due to the importance of the MVC problem, many researchers have instead focused their attention on the design of an approximation algorithm for delivering quality solutions in a reasonable time. One of the first approximation algorithms in the literature for the Unweighted Vertex Cover problem belongs to Gavril [5]. K. Clarkson has modified the greedy algorithm for the vertex cover problem [6]. Garey and Johnson have presented a simple approximation algorithm based on maximal matching that give an approximation ratio of 2 for the general graphs [7]. Recently, Ashay Dharwadker has presented a new polynomial-time algorithm to find minimal vertex covers in graphs [8]. Khuri and Back [9] have presented an evolutionary heuristic for the minimum vertex cover problem. For a comprehensive survey on the analysis of approximation algorithms for MVC, the reader is referred to Hochbaum [10], Monien and Speckenmeyer [11], Berman and Fujito [12], and Pullan [13].

In this paper, a fast algorithm called Isolation Algorithm (IA) has been proposed to solve the minimum vertex cover problem. The proposed algorithm isolates the vertex which has minimum degree and adds all vertices to vertex cover, which are adjacent to isolated vertex, in order to get a good solution. The effectiveness of the algorithm is shown by extensive computational experiments on DIMACS Benchmark Instances [14] and BHOSLIB instances for MVC [15]. The simulation results show that the IA can find the optimal solution or the nearest solution to the optimal solution at worst.

Section 2 describes the heuristic algorithm and their complexities. In Section 3, graph models used in the experiments are briefly described; some experiments and their results are mentioned. Section 4 summarizes and concludes the paper.

II. BASIC DEFINITION, ALGORITHM AND COMPUTATIONAL COMPLEXITY

A. Basic Definition

Neighborhood of a vertex: Let G=(V,E) be an undirected graph. Then for each $v \in V$, the neighborhood of v is defined by $N(v) = \{ u \in V \mid u \text{ is adjacent to } v\}$.

Degree of vertex: The degree of vertex $v \in V$, is denoted by d(v) and is defined by the number of neighbors of v.

Proposition: If there is a vertex whose degree is one, then its neighbor must be added to vertex cover.

B. The Proposed Isolation Algorithm (IA)

The proposed algorithm is designed to find the general minimum vertex cover of a graph G. The idea behind the algorithm is the *Proposition*. The algorithm always tries to create isolated vertices. If there is a vertex of degree one, the algorithm adds its neighbor to vertex cover; otherwise, it finds the minimum degree vertex and add sits all neighbors to vertex cover. Then, it updates the adjacency matrix of G by putting zero in to the row and column entries of the corresponding vertices in the Vertex Cover. The above process continues until the edge set E is empty. After finding an initial

solution, the algorithm tries to optimize the vertex cover set by removing redundant vertices from the solution. The pseudo-code of the isolation algorithm is given below.

Input: G(V,E).
Output: V_C of G where V_C is the minimum vertex cover of G.

1. $V_C \leftarrow 0$
2. **do**
3. Find the minimum degree vertex v_m
4. add all neighbors of v_m to the solution, $V_C = V_C \bigcup \{ N(v_m) \}$
5. delete all edges which are adjacent to vertices in $N(v_m)$
6. **while** ($E \neq \phi$)
7. scan the solution space for the redundant vertices
8. **if** v_i is in V_C and all vertices in $N(v_i)$ are in V_C as well **then** remove v_i from the solution
9. **end.**

Figure 1.The pseudo-code of the proposed algorithm.

C. Computational Complexity

The worst case complexity of the algorithm IA can be obtained as follows: Assume that G=(V,E) is a path graph. There are always vertices of degree one in path graph. In the proposed algorithm, it requires running time O(n) to find the minimum degree, so the algorithm can add just a vertex to the solution and delete all edges adjacent to this vertex respectively in running time O(n). After isolating the vertex, the graph is still a path graph. Thus, this procedure must be repeated n/2 times. Therefore, overall running time of the IA can be deduced as follows: $(n/2).(O(n)+O(n)) = O(n^2)$.

III. EXPERIMENTAL RESULTS & ANALYSIS

All the procedures of IA have been coded in C++ language. The experiments have been carried out an Intel Pentium Core2 Duo 2.6 GHz CPU and 2GB of RAM. The effectiveness of heuristic IA has been evaluated using 106 instances. These instances are divided into two sets. Simulations are carried out on two types of graphs: DIMACS benchmarks instances and BHOSLIB instances for MVC.

A. Results for DIMACS Benchmark Graphs

We have tested the proposed algorithm on DIMACS benchmark graphs with known results in order to analyze IA thoroughly [18]. These graphs are designed in terms of finding maximum cliques, so we have considered the clique benchmark graphs as $\overline{G}$.

In Table I and II, the first three columns show the type of the instances such as name, cardinality and density of the instances; the fourth one gives the optimum value, the fifth column denotes the solution value achieved by the proposed algorithm and the last two columns show running time and relative error of IA.

Table I and II, show that the proposed algorithm could reach the optimal solution for half of the DIMACS benchmark graphs. IA could find the nearest solution to the optimal solution at worst.

TABLE I
SIMULATION RESULTS FOR DIMACS BENCMARK GRAPHS

$\overline{G}$	|V|	Density	Optimum V_c	IA	Time(s)	Relative Error
brock200_1	200	0.745	179	180	0.000	0,559
brock200_2	200	0.496	188	191	0.000	1,596
brock400_1	400	0.748	373	378	0.000	1,340
brock400_2	400	0.749	371	378	0.000	1,887
brock800_1	800	0.649	777	781	0.032	0,515
brock800_2	800	0.651	776	781	0.438	0,644
C125.9	125	0.898	91	91	0.000	0,000
C250.9	250	0.899	206	207	0.094	0,485
C500.9	500	0.9	443	447	0.016	0,903
C1000.9	1000	0.901	932	941	0.047	0,966
C2000.5	2000	0.5	1984	1987	0.219	0,151
C4000.5	4000	0.5	3982	3985	0.969	0,075
c-fat200-1	200	0.077	188	188	0.000	0,000
c-fat200-2	200	0.163	176	176	0.000	0,000
c-fat200-5	200	0.426	142	142	0.000	0,000
c-fat500-1	500	0.036	486	486	0.000	0,000
c-fat500-2	500	0.073	474	474	0.000	0,000
c-fat500-5	500	0.186	436	436	0.000	0,000
c-fat500-10	500	0.374	374	374	0.000	0,000
DSJC500.5	500	0.5	487	487	0.282	0,000
DSJC1000.5	1000	0.5	985	987	0.062	0,203
gen200_p0.9_44	200	0.9	156	156	0.109	0,000
gen200_p0.9_55	200	0.9	145	145	0.015	0,000
gen400_p0.9_65	400	0.9	345	350	0.015	1,449
gen400_p0.9_75	400	0.9	325	325	0.640	0,000
hamming6-2	64	0.905	32	32	0.000	0,000
hamming6-4	64	0.349	60	60	0.000	0,000
hamming8-2	256	0.969	128	128	0.000	0,000
hamming8-4	256	0.639	240	240	0.000	0,000
hamming10-2	1024	0.99	512	512	0.063	0,000
hamming10-4	1024	0.829	984	989	0.078	0,508
johnson8-2-4	28	0.556	24	24	0.000	0,000
johnson8-4-4	70	0,226	56	56	0.000	0,000
johnson16-2-4	120	0,235	112	112	0.000	0,000
johnson32-2-4	496	0,121	480	480	0.000	0,000
keller4	171	0.350	160	160	0.000	0,000
keller5	776	0,248	749	752	0.453	0,401

TABLE II
SIMULATION RESULTS FOR DIMACS BENCMARK GRAPHS

$\overline{G}$	|V|	Density	Optimum V_c	IA	Time(s)	Relative Error
keller6	3361	0.818	3302	3316	0.813	0,424
MANN_a9	45	0.927	42	42	0.000	0,000
MANN_a27	378	0.99	375	375	0.000	0,000
MANN_a45	1035	0.996	1032	1032	0.047	0,000
MANN_a81	3321	0.999	3318	3318	0.453	0,000
p_hat300-1	300	0.244	292	292	0.062	0,000
p_hat300-2	300	0.489	275	276	0.000	0,364
p_hat300-3	300	0.744	264	266	0.203	0,758
p_hat500-1	500	0.253	491	492	0.016	0,204
p_hat500-2	500	0.505	464	465	0.032	0,216
p_hat500-3	500	0.752	450	452	0.016	0,444
p_hat700-1	700	0.249	689	692	0.031	0,435
p_hat700-2	700	0.498	656	657	0.062	0,152
p_hat700-3	700	0.748	638	638	0.063	0,000
p_hat1000-1	1000	0.245	990	991	0.937	0,101
p_hat1000-2	1000	0.49	954	954	0.140	0,000
p_hat1000-3	1000	0.744	934	936	0.157	0,214
p_hat1500-1	1500	0.253	1488	1491	0.125	0,202
p_hat1500-2	1500	0.506	1435	1435	0.438	0,000
p_hat1500-3	1500	0.754	1406	1412	0.360	0,427
san200_0.9_1	200	0.9	130	130	0.125	0,000
san200_0.9_2	200	0.9	140	140	0.360	0,000
san200_0.9_3	200	0.9	156	156	0.109	0,000
san400_0.7_1	400	0.7	360	360	0.000	0,000
san400_0.9_1	400	0.9	300	300	0.000	0,000
san1000	1000	0.502	985	990	0.047	0,508
sanr200_0.7	200	0.697	182	183	0.000	0,549
sanr400_0.5	400	0.501	387	388	0.125	0,258
sanr400_0.7	400	0.7	379	380	0.000	0,264

B. Results for BHOSLIB Graphs

We also have tested the proposed algorithm on BHOSLIB instances for MVC which are generated with RB Model [16]. This type of graph is generated as follows. Generate n disjoint cliques, each of them has n^a (where a>0 is a constant) and select two different cliques randomly, then add edges between these cliques randomly. These instances are the most difficult graph types to find the minimum vertex cover. IA could find near optimal solutions even for these instances.

Table III, has the same template with Table I and II except for the density column.

TABLE III
SIMULATION RESULTS FOR BHOSLIB GRAPH INSTANCE

G	|V|	Optimum V_c	IA	Time(s)	Relative Error
frb30-15-1	450	420	424	0.016	0,952
frb30-15-2	450	420	423	0.031	0,714
frb30-15-3	450	420	424	0.016	0,952
frb30-15-4	450	420	422	0.031	0,476
frb30-15-5	450	420	424	0.015	0,952
frb35-17-1	595	560	565	0.016	0,893
frb35-17-2	595	560	565	0.047	0,893
frb35-17-3	595	560	564	0.063	0,714
frb35-17-4	595	560	564	0.046	0,714
frb35-17-5	595	560	565	0.031	0,893
frb40-19-1	760	720	725	0.063	0,694
frb40-19-2	760	720	725	0.078	0,694
frb40-19-3	760	720	724	0.031	0,556
frb40-19-4	760	720	725	0.078	0,694
frb40-19-5	760	720	723	0.046	0,417
frb45-21-1	945	900	907	0.062	0,778
frb45-21-2	945	900	904	0.063	0,444
frb45-21-3	945	900	906	0.124	0,667
frb45-21-4	945	900	905	0.109	0,556
frb45-21-5	945	900	906	0.062	0,667
frb50-23-1	1150	1100	1108	0.140	0,727
frb50-23-2	1150	1100	1108	0.094	0,727
frb50-23-3	1150	1100	1105	0.109	0,455
frb50-23-4	1150	1100	1108	0.094	0,727
frb50-23-5	1150	1100	1106	0.094	0,545
frb53-24-1	1272	1219	1226	0.125	0,574
frb53-24-2	1272	1219	1224	0.141	0,410
frb53-24-3	1272	1219	1226	0.125	0,574
frb53-24-4	1272	1219	1227	0.109	0,656
frb53-24-5	1272	1219	1227	0.109	0,656
frb56-25-1	1400	1344	1352	0.156	0,595
frb56-25-2	1400	1344	1353	0.156	0,670
frb56-25-3	1400	1344	1353	0.156	0,670
frb56-25-4	1400	1344	1353	0.218	0,670
frb56-25-5	1400	1344	1352	0.265	0,595
frb59-26-1	1534	1475	1481	0.234	0,407
frb59-26-2	1534	1475	1483	0.374	0,542
frb59-26-3	1534	1475	1483	0.359	0,542
frb59-26-4	1534	1475	1484	0.172	0,610
frb59-26-5	1534	1475	1482	0.203	0,475

IV. CONCLUSION

A new algorithm IA for MVC of a graph has been proposed, and its performance has been investigated. Simulation experiments have shown the effectiveness of the algorithm. The experimental results have also shown that this simple approach is extremely fast. Furthermore, it has been seen that the algorithm can find optimal solutions for more than half of DIMACS benchmark graph instances. The algorithm has given near optimal solutions for most of the BHOSLIB instances for MVC in reasonable times.

ACKNOWLEDGMENT

The author would like to thank to Urfat Nuriyev, Murat Ersen Berberler, Gozde Kizilates and Asli Guler for many helpful discussions and comments.

REFERENCES

[1] H.Cormen T. H., Lieserson C. E., Rivest R. L. and Stein C., Introduction to Algorithms, Second Edition, The MIT Press, 2001.

[2] R. M. Karp, Reducibility among combinatorial problems, Plenum Press, New York, (1972), pp 85 - 103.

[3] M.Garey and D. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, Freeman, San Francisco, 1979.

[4] U.Stege, Resolving conflicts from problems in computational Biology, Ph. D thesis, No.13364, ETH Zurich (2000).

[5] Gavril F. Private Communication cited in [16], 1974.

[6] K. Clarkson, A modification to the greedy algorithm for the vertex cover problem‖, IPL, Volume 16:23-25, (1983).

[7] M.R.Garey, D. S. Johnson, Computers and Intractability: A Guide to the theory NP - completeness, San Francisco: Freeman (1979).

[8] Dharwadker A. , "The Vertex Cover Algorithm", Createspace, 2011.

[9] S. Khuri and T. Back, An evolutionary heuristic for the minimum vertex cover problem, J. Kunze and H. Stoyan, editors, KI - 94 workshops (Extended Abstracts), Bonn (1994), pp. 83 – 84.

[10] D.S.Hochbaum Efficient bounds for the stable set, vertex cover and set packing problems, Discrete Appl. Mathematics, Vol. 6, (1983), 243 - 254.

[11] B.Monien and E. Speckenmeyer Ramsey numbers and an approximation algorithm for the vertex cover problems, Acta Informatica, Vol. 22, (1985), 115 - 123.

[12] P.Berman and T. Fujito, On approximation properties of the independent set problem for low degree graphs, Theory of Computing Syst., Vol. 32, (1999), 115 - 132.

[13] W.Pullan, Optimization of unweighted/weighted maximum independent sets and minimum vertex covers, Discrete Optimization, Vol. 6, (2009), 214-219.

[14] DIMACS clique benchmarks, Benchmark instances made available by electronic transfer at dimacs.rutgers.edu, Rutgers Univ., Piscataway. NJ. (1993).

[15] BHOSLIB instances for vertex cover available at: www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm

[16] Xu K. Boussemart F., Hemery F, Lecoutre C., A Simple Model to Generate Hard Satisfiable Instances, IJCAI'05 Proceedings of the 19th international joint conference on Artificial intelligence, USA, (2005).