Vertex Cover Problem—Revised Approximation Algorithm

Kartik Shah, Praveenkumar Reddy and R. Selvakumar

Abstract This paper is aimed to present the solution to vertex cover problem by means of an approximation solution. As it is NP complete problem, we can have an approximate time algorithm to solve the vertex cover problem. We will modify the algorithm to have an algorithm which can be solved in polynomial time and which will give near to optimum solution. It is a simple algorithm which will be based on articulation point. Articulation point can be found using the Depth First Search algorithm.

Keywords Articulation point · Vertex covering problem · Optimization · Approximation algorithm

1 Introduction

A problem (*P*) is called NP complete if *P* is in class NP and every problem *P′* is polynomially reducible to *P*. Vertex cover problem is that problem in which we take an undirected graph ($G = (V, E)$) with '*V*' vertices and '*E*' edges and will have a set which contains the vertices which can cover all the edges of the graph. According to Garey and Johnson, vertex cover problem is one of the six basic NP complete problems [1].

It is observed that sometimes, there exist some problems for which we do not have an optimal solutions. However, we can have an approximate solution which

K. Shah (✉) · P. Reddy · R. Selvakumar
School of Computing Science and Engineering, VIT University, Vellore 632014, India
e-mail: kartikshah@mail.com

P. Reddy
e-mail: praveenkumar.reddym2012@vit.ac.in

R. Selvakumar
e-mail: rselvakumar@vit.ac.in

L.P. Suresh et al. (eds.), *Artificial Intelligence and Evolutionary Algorithms in Engineering Systems*, Advances in Intelligent Systems and Computing 324,
DOI 10.1007/978-81-322-2126-5_2

will be closer to the optimal solution. The algorithm which provides closer solution to the optimal solution is known as approximation algorithms.

Let $G = (V, E)$ be a graph where V is number of vertices and E be number of edges. The set of vertices $V' \subseteq V$ is said to be cover if for each edge $(u, v) \in E$, either $u \in V'$ or $v \in V'$ or both. The number of vertices in V' is known as the size of the cover V'. The problem of finding minimum size is known as vertex cover problem.

2 Related Work

For vertex cover problems, we have many real-life problems in which vertex cover problem solution can be applied. Like, to find population growths taken into polynomial time and for that we can take bi-parted graph and also take articulation points [2]. Also, vertex cover problem for network base routing delays in tolerance network [3], for network traffic measurement [4]. Polynomial space parameterized vertex cover can be solved in $O(1.2738^k + kn)$ time [5]. F. Delbot and C. Laforest had proposed an algorithm in which vertexes are scanned from left to right. Condition is as follows: "*u* is added to vertex cover if and only if it has at least one neighbor not in the cover" [6]. This algorithm is called as LIST LEFT which is given as follows [7]:

```
Labeled graph $L(G) = (L(V), E)$
1. $C \leftarrow \phi$
2. For each vertex, $u \in L(V)$ do
3. If $u$ has at least one right neighbor, then
4. $C \leftarrow C \cup \{u\}$
5. Return $C$
```

Sorted LL is another technique of finding vertex cover. It takes decision as "if there is at least one $v \in N(u)$ with lower degree, select u; otherwise, if u has only neighbor with higher degree, u is not selected." Algorithm can be represented as follows [7]:

```
Labeled graph $L(G) = (L(V), E)$
1. $C \leftarrow \phi$
2. For each vertex, $u \in L(V)$ do
3. If $u$ has at least one right neighbor with a lower degree or a right neighbor with
   the same degree, then
4. $C \leftarrow C \cup \{u\}$
5. Return $C$
```

One more approach is to use ANTI SORTED LL. In this algorithm, degree of vertex's neighbor is taken into consider. Below is the algorithm shown [7]:

Labeled graph $L(G) = (L(V), E)$

```
1. $C \leftarrow \phi$
2. For each vertex, $u \in L(V)$ do
3. If $u$ has at least one right neighbor with a larger degree or a left neighbor with
   the same degree, then
4. $C \leftarrow C \cup \{u\}$
5. Return $C$
```

These all algorithms can be used for finding vertex cover set of a graph.

It is possible to approximate the weight of the vertices which results in local approximation and called as local-ratio theorem and is so called local-ratio theorem for weighted vertex cover problem [8]. Further, it states that it is possible to put together the Nemhauser–Trotter algorithm (local optimization algorithm) and local-ratio theorem to get new approximation techniques that improve performance [8]. Kernelization is the process of applying polynomial time preprocessing to the instance of graph $G(V, E)$ and obtaining other instance $G'(V', E')$ where $V' \leq V$ and G' will have vertices of vertex cover V' [9]. It is observed that we can find up to 3/2 vertices of minimum vertex covers using collection of graph transformations. The algorithm guarantees an approximation ratio of 3/2, for finding large number of randomly created graphs. The reductions are extremely fast. The problem has best-case and worst-case approximation ratio as 2-O (1) [10]. The random graphs that we generally use are simple random graph model proposed by y Erdos and Renyi [11]. Chromatic number is defined as minimum number of colors used for vertex to cover all with distance 1. Chromatic number can be used to find the vertex over of the graph. Kuhn and Mastrolilli [12] investigated weighted vertex cover problem for graphs when a locally bounded coloring is given.

3 Approach

There are many approaches to solve vertex cover problem.

3.1 Classical Approach

The approximation algorithm which is available to solve the vertex cover problem is a polynomial time algorithm, and it can be solved in $O\ (V + E)$ time complexity where V is number of vertices and E is number of edges. The approximation vertex cover is a polynomial time 2 approximation algorithm.

That is, let V be a vertex cover and V^* be the optimal cover to that problem. Then, we can prove that

$$|V| \leq 2|V^*|$$

So, in this case, if number of vertices increases, the solution to vertex cover may also diverse far from optimal solution.

Algorithm 1.1 Approximation Algorithm [13]

```
APPROX VERTEX COVER($G$)
1. $C \leftarrow \phi$
2. $E' = E[G]$
3. While $E' \leftarrow \phi$
4. Let $(u, v)$ be an arbitrary edge of $E'$
5. $C \leftarrow C \cup \{u, v\}$
6. Remove from $E'$ every edge incident on either $u$ or $v$
7. Return $C$
```

3.2 Our Approach

As we noted above that approximate vertex cover problem solution is less than or equal to 2 times the optimal solution. We proposed the algorithm which will take near to optimal solution.

In our approach, we will find the articulation point of that particular graph and then add those vertices in the vertex cover. From remaining edges, we will take the common vertex which can cover 2 or more edges. After that if some more edges are not covered, then we will take 1 of the vertex from the edge. In order to find the articulation point, we can go for Depth First Search (DFS) algorithm, which takes $O\ (V + E)$ time to find the articulation points. There are many more algorithms available for finding articulation points. DFS is one of the simplest algorithms.

Proposed algorithm to find the vertex cover of graph is shown below.

Algorithm 1.2 Proposed Vertex Cover Algorithm

```
1. $C \leftarrow \phi$
2. $C'$ = ArticPointDFS (vertex);
3. $C = \leftarrow C\ U\ C'$
4. Repeat Step 2 and Step 3 until all the Articulation Point is found;
5. $E' = E\ [G]$;
6. $E' = E' - \{$set of edges covered by $C'\}$
7. While $E' \neq \phi$
Let $(a, b)$ be an arbitrary edge, check $(a, b) \cap \{$take each edge of $E'\} \neq \phi$ Then,
   add any of vertices '$a$' or '$b$' in $C$.
Remove edge $(a, b)$ from $E'$
Else add the common vertex.
Remove edges containing that vertices from $E'$
8. Return $C$.
```

4 Analysis

Now comparing both the algorithms, let us take 1 case of finding vertex cover. Consider the following example. It contains 10 vertices and 13 edges. All vertices are connected with edges as shown in Fig. 1. Now, we will apply the vertex cover algorithms to the following example with both the approaches (i.e., existing approach and proposed approach). After that result will be compared for example, and output from each algorithm will be compared. The algorithm is computed step by step as per the algorithms discussed above.

Using existing algorithm:

1. $C \leftarrow \phi$
2. E' = {(1, 2), (2, 3), (3, 4), (1, 4), (2, 7), (2, 9), (2, 10), (3, 5), (3, 6), (7, 8), (7, 9), (7, 10), (9, 10)}
3. Let us take an arbitrary edge (9, 10)
4. C = {9, 10}
5. E' = {(1, 2), (2, 3), (3, 4), (1, 4), (2, 7), (2, 9), (2, 10), (3, 5), (3, 6), (7, 8)}
6. Let us take another arbitrary edge (1, 4)
7. C = {1, 4, 9, 10}
8. E' = {(2, 3), (2, 7), (2, 9), (2, 10), (3, 5), (3, 6), (7, 8)}

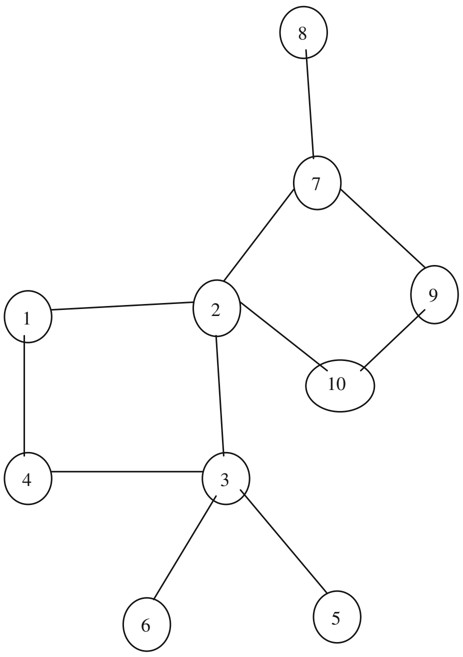


Fig. 1 Vertices are connected with each other, and a complex *graph* is formed

9. Let us take another arbitrary edge (3, 5)
10. C = {1, 3, 4, 5, 9, 10}
11. E' = {(2, 7), (7, 8)}
12. Let us take another arbitrary edge (7, 8)
13. C = {1, 3, 4, 5, 7, 8, 9, 10}

Finally, vertex cover contains 8 elements.
Using proposed algorithm:

1. $C \leftarrow \phi$
2. E' = {(1, 2), (2, 3), (3, 4), (1, 4), (2, 7), (2, 9), (2, 10), (3, 5), (3, 6), (7, 8), (7, 9), (7, 10), (9, 10)}
3. Find articulation points, C' = {2, 3, 7}
4. C = {2, 3, 7}
5. $E' = E' -$ {set of edges covered by C'}
6. E' = {(1, 4), (9, 10)}
7. Take (1, 4) as an arbitrary edge.
8. C = {1, 2, 3, 7}
9. E' = {(9, 10)}
10. Take (9, 10) as an arbitrary edge.
11. $E' = \phi$
12. C = {1, 2, 3, 7, 9}

Finally, vertex cover contains 5 elements.

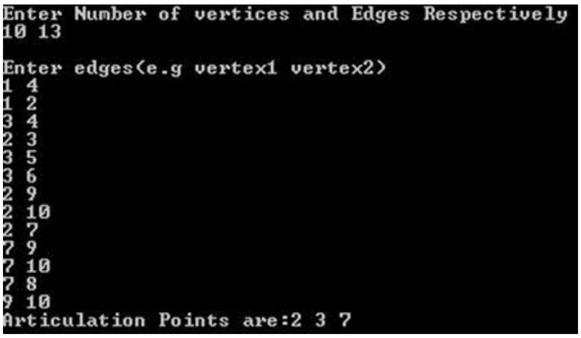
As we can see the result from existing approach and from proposed approach, it is clear that the existing approach can give us the true result but that will not be minimal. Some vertices that are not needed will also be included in the vertex cover list, while the proposed approach will have less number of vertices in the vertex cover list and which covers all the edges. Using this approach, first we will collect all the articulation point details and will include that vertices into vertex cover list. After that we will go with normal approach for remaining vertices, and the result will be as shown in above example. The implementation of the above approach is discussed in following section.

5 Implementation

Implementation of proposed algorithm can be done in 2 steps:

Step 1: Find the articulation points of the graph given using articulation point algorithm (Using DFS algorithm). Add articulation points in the set of vertex cover. For example, given above, after applying algorithm, we can get following output (Fig. 2).

Fig. 2 Step 1 of implementation



```
Enter Number of vertices and Edges Respectively
10 13

Enter edges(e.g vertex1 vertex2)
1 4
1 2
3 4
2 3
3 5
3 6
2 9
2 10
2 7
7 9
7 10
7 8
9 10
Articulation Points are:2 3 7
```

Step 2: Remove the edges which are adjacent to all the vertices listed in Step 1. Apply approx vertex cover algorithm on remaining graph (Disconnected Graph). For example, given above, we have 4 choices {(1, 9) or (1, 10) or (4, 9) or (4, 10)}. We can choose any combination.

Step 3: Take union of vertices found in Steps 1 and 2.

6 Conclusion

From above example, we can see that using existing algorithm, we are getting 8 elements in vertex cover set. While using proposed algorithm, we are getting 5 elements. Also, it is same as minimal vertex cover.

If V is a vertex cover set derived from Algorithm 1.1 and V^* is an optimal vertex cover set, then

$$|V| \le 2|V^*|$$

If C is a vertex cover set derived from Algorithm 1.3 and C^* is an optimal vertex cover set, then

$$|C| \approx |C^*|$$

Proposed algorithm takes O (2 ($V + E$)) time complexity. O ($V + E$) for DFS and O ($V + E$) for finding vertex cover. So, total O ($V + E$) time to compute vertex cover is same as the available algorithm, but our algorithm provides much nearer solution to optimal solution.

7 Future Work

In this proposed algorithm, we are using DFS algorithm to find the articulation points of the graph. This algorithm takes $O\ (V + E)$ time to compute the articulation points. We can go for some other techniques to find the articulation points which can take less time compared to DFS. Also, we can find some other techniques which always provide the exact solution to optimal solution. Also, some more techniques can be used to provide nearer solution. There is also possible to find algorithm which runs faster than this algorithm.

Acknowledgment We would like to thank the School of Computer Science and Engineering, VIT University, for giving us such an opportunity to carry out this research work and also for providing us the requisite resources and infrastructure for carrying out the research.

References

1. M. Garry, D. Johnson, *Computers and Intractability: A User Guide to the Theory of NP Completeness* (San Francisco, 1979)
2. P.S. Oliveto, X. Yao, J. He, Analysis of Population-based Evolutionary Algorithms for the Vertex Cover Problem (IEEE, 2008), pp. 1563–1570
3. L. Ding, B. Gu, X. Hong, B. Dixon, Articulation node based routing in delay tolerant networks, in IEEE International Conference (2009), pp. 1–6
4. Y. Zeng, D. Wang, W. Liu, A. Xiong, An approximation algorithm for weak vertex cover problem in IP network traffic measurement, IEEE International Conference (2009), pp. 182–186
5. J. Chen, I.A. Kanj, G. Xia, Improved parameterized upper bounds for vertex cover. 31st International Conference on Mathematical Foundations of Computer Science (2006)
6. F. Delbot, C. Laforest, A better list heuristic for vertex cover. Inf. Process. Lett. **107**, 125–127 (2008)
7. E. Angel, R. Campigotto, C. Laforest, Algorithm for the vertex cover problem on large graphs. IBISC Research report (2010)
8. R. Bar-Yehuda, S. Even, A local-ratio theorem for approximating the weighted vertex cover problem. Ann. Discrete Math. **25**, 27–46 (1985)
9. G.L. Nemhauser, L.E. Trotter, Vertex packing: structural properties and algorithms. Math. Program. **8**, 232–248 (1975)
10. E. Asgeirsson, C. Stein, *Vertex Cover Approximations on Random Graphs* (Springer, Berlin, 2007), pp. 285–296
11. P. Erdos, A. Renyi, On random graphs. Publ. Math. Debrecen **6**, 290–297 (1959)
12. F. Kuhn, M. Mastrolilli, Vertex cover in graphs with locally few colors (2011), pp 498–509
13. D. Hochbaum, Approximation algorithms for the set covering and vertex cover problems. SIAM J. Comput. **11**(3), 555–556 (1982)

Springer

http://www.springer.com/978-81-322-2125-8

Artificial Intelligence and Evolutionary Algorithms in Engineering Systems
Proceedings of ICAEES 2014, Volume 1
Suresh, L.P.; Dash, S.S.; Panigrahi, B.K. (Eds.)
2015, XVII, 862 p. 376 illus., Softcover
ISBN: 978-81-322-2125-8