Performance Evaluation of Vertex Cover and Set Cover Problem using Optimal Algorithm

B. M. Monjurul Alom, Someresh Das and Mohammad Abdur Rouf

Dept. of CSE, Dhaka University of Engineering & Technology (DUET), Gazipur, Bangladesh

E-mail: bm.alom@uon.edu.au

ABSTRACT

An approximation algorithm for a procedure that always provides some kind of solution, although it may fail to find the optimal solution. The approximate algorithms generally faster which can be proved to be close to the optimal solutions. The existing approximate algorithm for vertex cover and set cover problem is not optimal. To overcome this limitation, in this paper we have presented two new algorithms for vertex cover and set cover problem that provides the optimal solution that is better than approximate solution. The presented optimal algorithm and the existing well known approximation algorithm have almost the same time complexity but with respect to the solution our optimal algorithm outperforms compared to the existing approximation algorithm.

1. INTRODUCTION

If we try to identify those contributions of computer science which will be long lasting, surely one of these will be the refinement of the concept called algorithm. Informally an algorithm is any well-defined computational procedure that takes some value or set of values as input and produces some value or values as output. An algorithm is thus a sequence of computational steps that transform the input into the output. An approximation algorithm may fail to find the optimal solution but always provides some kind of solution to specific problem. The approximate algorithms generally faster which can be proved to be close to the optimal solutions. The existing approximate algorithm for vertex cover and set cover problem is not optimal. To overcome this limitation, we have developed two new optimal algorithms for vertex cover and set cover problem that provides the optimal solution that is better than approximate solution.

Consider a graph G = (V, E) where V and E are accordingly vertex and edges. A vertex cover of an undirected graph is a subset $V' \leq V$ such that if (u, v) is an edge of G. Then either $u \in V'$ or $v \in V'$ or both. The size of a vertex cover is the number of vertices in it. The vertex cover problem is to find a vertex cover of minimum size in a given undirected graph. Such a vertex cover is called an optimal vertex cover. An instance (X, F) of the set covering problem consists of a finite set X and a family F of subsets of X, such that every element of X belongs to at least one subset in F.

$$X = \bigcup_{S \in F} S$$

We say that a subset S Є F converting element. The problem is to find a minimum size subset C Є F where member cover all of X.

$$X = \bigcup_{S \in C} S$$

Optimal vertex cover algorithm selects the vertex which has maximum number of edges incident to it. All the edges are discarded incident to that vertex. If more than one vertex have same maximum number of edges, this algorithm select that vertex which have at least one edge that is not covered by other vertices which has maximum edge. This process is repeated until to cover all the vertices of the graph. Optimal set cover algorithm selects the set which has maximum number of elements among all the sets. Then the set is subtracted from all the sets. If more than one set have maximum number of elements, this algorithm select the set which have more element in the previous state. This process is continued to cover all the set. In the final set, if all the elements of any set is covered by other sets, that set is discarded to get the optimal set cover.

In this paper we have described related work in section 2, existing algorithm of vertex cover is presented in section 3, our proposed optimal algorithm of vertex cover is in section 4, existing algorithm of set cover problem is in section 5, and optimal algorithm of set cover problem is in section 6, experimental evaluation in section 7. The paper concludes in section 8.

2. RELATED WORK

The An embellished greedy algorithm given in [1] *SaGaR* (short for Sort-and-Greedy-and-Remove) for set cover problem . The time complexity of *SaGaR* is proved to be the same as that of the well-known greedy algorithm (*Greedy*). The algorithm *SaGaR* is a practical method and may function much better in practice than the algorithm

Greedy, especially for problems with some special structure. A greedy algorithm is given for set cover in [2]. This greedy algorithm doesn't find optimal solution. An approximation algorithm for vertex cover is given in [2]. This algorithm finds the approximate solution. An algorithm for "crown reductions for the Minimum Weighted Vertex Cover problem" is given in [3]. In this algorithm how crown decompositions and strong crown decompositions can be computed in polynomial time are described. An efficient amount of research have been done on set cover and vertex cover problem are described in [1, 3-18].

3. EXISTING APPROXIMATION ALGORITHM

The existing algorithm is given in Fig 1:

```
Approximate_vertex_cover (G) {
1. C ← ϕ
2. E/ ← E[G]
3. while E/ ≠ ϕ
4. do let (u,v) be an arbitrary edge
   of  E/
5. C ← C U {u,v}
6. remove from E/ every edge incident
   on either u or v
7. Return C    }
```

Fig. 1: Approximate_vertex_cover.

3.1 Explanation of the algorithm with example

Considering the Fig. 2, the solution of the graph according to the approximate algorithm is {b, c, d, e, f, g}. This algorithm selects a vertex arbitrary and removes the incident edges. This process continues until to cover all the vertex. At first b is chosen from Fig. 2, and then ba , bc edges are discarded from the graph. Next c is chosen, ce and cd edges are removed. Now the graph contains only d, e, f, g vertexes. Similarly d, e, f and g vertexes are chosen and hence all the vertexes are covered.

3.2 Complexity analysis of the algorithm

Since the loop on lines (3-6) repeatedly picks an edge (u,v) from $E^{/}$, adds its endpoints u and v to C, and deletes all edges in $E^{/}$ that are covered by either u or v. The running time of this algorithm is O(E).

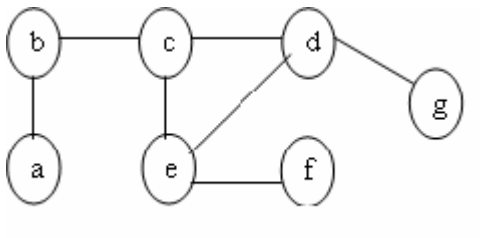

Fig. 2: Undirected graph.

4. OPTIMAL ALGORITHM OF VERTEX COVER PROBLEM

```
OPTIMAL_VT_COVER (E, V) {
  // E is an edge and V is an vertex
1.   V′ ← ϕ;
2.   E/ ← E[G]
3.    While (E/ ≠ ϕ) {
4.  M   ←     vertex which has maximum
   incident edge;
5. If (More than one vertex have maximum
   number of edges) then
     M ← Choose that node which has at
     least one edge that is not covered
     by others which have maximum number
     of edges.
6. V′  ←  V′ U M;
7.  Remove the all incident edges at
   vertex M;
8. Count incident edge of new graph.}
9. Return V′ }
```

Fig. 3: Optimal vertex cover algorithm.

4.1 Explanation of the Algorithm

Step1: Counting incident edges of all vertices in Fig. 2, we see a=1, b=2, c=3, d=4, e=3, f=2, g=1.

Step2: We find d has the maximum edges it is 4. Now discard all the edges incident to d given in Fig. 4.

Step 3: Again counting the edges of remaining vertices, we have a=1, b=2, c=2, e=2 in Fig. 4. Here maximum edges are 2 and it has in vertex b, c, and e. c has two edges that is covered by b and e. Now b and e both have two edges but they have at least one edge that is not covered by other vertices c, which has maximum edge, this algorithm select either b or e. Here we have chosen b. Remove all the edges incident to b we have only c and e given in Fig. 5.

Step 4: Counting edges of c and e in Fig. 5, c=1, e=2.Now selects e and the final graph is given in Fig. 6. So the optimal vertices covering set is {b, d, e}.

4.2 Complexity Analysis of the Vertex Cover Algorithm

Since the number of iterations of the loop on lines 3-8 is at most E. So time complexity of this algorithm is O (E).

4.3 Comparison between Approximation Vertex Cover and Optimal Vertex Cover Algorithm

In the approximate algorithm in each step add one or more edges, where in the optimal algorithm add only one vertex at a time which take less memory than storing edges. The solution of above graph according to the approximate algorithm is {b, c, d, e, f, g} where the solution according to optimal algorithm is {b, d, e}.

4.4 The Size of an Optimal Cover

Since this a minimization problem, we are interested in smallest possible c/c^*. Specifically we want to show $c/c^* \leq 2 = p(n)$. In other words, we want to show that APPROX-VERTEX-COVER algorithm returns a vertex-cover that is atmost twice the size of an optimal cover.

Proof: Let the set c and c^* be the sets output by APPROX-VERTEX-COVER and OPTIMAL-VERTEX-COVER respectively. Also, let A be the set of edges selected by line 4. Because, we have added both vertices, we get $c = 2|A|$ but OPTIMAL-VERTEX-COVER would have add one of two => $c/c^* \leq p(n) = 2$. Formally, since no two edge in A are covered by the same vertex from c^* (since, once an edge is picked in line 4, all other edges that are incident on its endpoints are deleted from E` in line 6) and we the lower bound:

$$|c^*| \geq A \quad (1)$$

On the size of an OPTIMAL-VERTEX-COVER. In line 4 of Fig. 1, we picked both en points yielding an upper bound on the size of Vertex-Cover. $|c| \leq 2|A|$ since, upper bound is an exact in this case, we have

$$|c| = 2|A| \quad (2)$$

Take $|c|/2 = |A|$ and put it in Eqn. (1)

$|c^*| \geq |c|/2$

$|c^*|/|c| \geq ½$

$|c^*|/|c| \leq 2 = p(n)$.

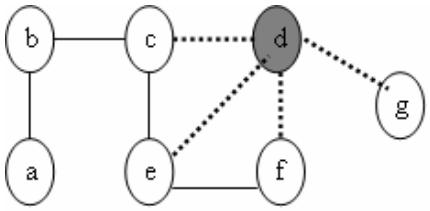

Fig. 4: Discarding the edges incident to vertex d.

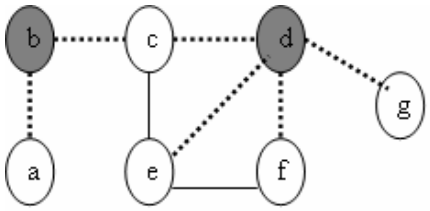

Fig. 5: Discarding the edges incident to vertex b.

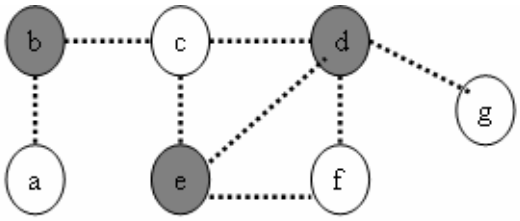

Fig. 6: Graph represents the optimal vertex covering set.

5. EXISTING ALGORITHM OF SET COVER

```
The algorithm is given in Fig.7.
 Algorithm Greedy_Set_cover(X, F)
//  X  is  the  number  of  element  in
universal set
1.          S=0;
// F is subsets
2.  While (X! =0) {
3.    S'= find the subject which has
maximum element;
4.    S=S U S';
5.    F=F-S;
6.    X=X-element of S';
          } }
```

Fig. 7: Algorithm set_cover.

5.1 Explanation of the Set_Cover Algorithm

Considering the Fig. 8. An instance (X, F) of the set covering problem, where the sets consist of the 12 black point and F= {S1, S2, S3, S4, S5, S6}. According the algorithm the first set S1 will be selected as it has maximum number of elements. Subtracting this element to the original set, S4 has the maximum number of elements. So S4 will be selected. Similarly S5 and S3 will be selected. Hence the set cover will be { S1, S4, S5, S3}.

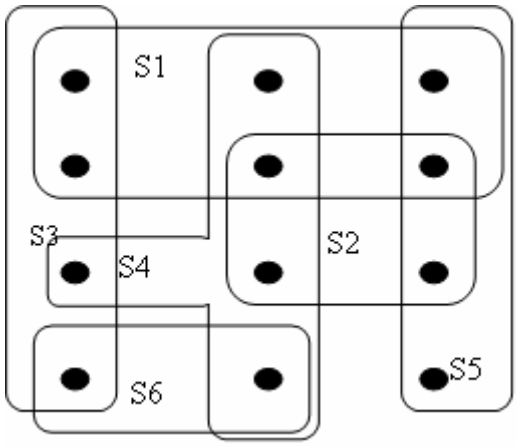

Fig. 8: Six Sets consist of different blocks.

5.2 Time Complexity of Set Cover Algorithm

We Since the number of iterations of the loop on lines 2-6 is at most min (|X|, |F|). And the loop body runs in time O(|X| |F|).Total implementation run time is: O(|X||F| min(|X|,|F|)).

6. PROPOSED ALGORITHM FOR MINIMUM SET COVER

```
Algorithm Min_Set-cover(X, F)      // X
is the number of element in universal
set
1. S ← ϕ;              // F is subsets
2. While (x′ ≠ ϕ) {
```

```
3. S'= find the subset which has maximum
   element;
4. If (more than one subset has maximum
   element) then
5. Choose that subset which has more
   elements in previous set among them
6. S=S U S';
7. F=F-S;
8. X=X-element of S';      }
9. For each set of S'
10. If (all the element of any set (say
     P) of S' is covered by other subset
     of S') then
11. S'=S'-P;

    Return S'
}
```

Fig. 9: Minimum Set Cover Algorithm.

6.1 Explanation of the Minimum Set_Cover Algorithm

Considering the given Fig. 8.

Step1: Set S1 has maximum element and it is 6. Select it and store in S.

Step2: Subtract S1 from S1, S2, S3, S4, S5 and S6

Step3: Now S1=0 S2=2 S3=2 S4=3 S5=2 S6=2. So we select S4 as it has maximum element.

Step4: Aging subtract S4 from S2, S3, S4, S5 and S6 and the number of element of sets become S1=0, S2=1, S3=1, S4=0, S5=2, S6=2

Step5: Here maximum element in 2 and it is in S5 also in S6; here choose S5 because it has 4 elements in the previous set. After that the sets become S1=0, S2=0, S3=1, S4=0, S5=0, S6=1. And in the same way select S3 from remaining set.

Step6: We get S1, S4, S5 and S3 as an approximate set cover. We see all the elements of S1 is covered by S4, S5 and S3 so we can discard S1 and Minimum set Cover Size is 3 and that is {S3, S4,S5}.

6.2 Time Complexity of the Minimum Set Cover Algorithm

Since the number of iterations of the loop on lines 2-8 is at most min (|X|, |F|). And the loop body runs in time O(|X| |F|). Total implementation run time is: O(|X||F| min (|X|,|F|)). There is no extra time is necessary more than this to execute on the last two lines 9-11.

6.3 Comparison Analysis between Existing and Proposed Algorithm

The greedy algorithm given in section 5, produces a cover of size 4 using Fig. 8 by selecting the sets S_1,S_4,S_5 and S_3 . But our minimum set cover algorithm produces S_3, S4 and S_5. But the time complexity is almost same. Our algorithm is better in case of solution.

7. EXPERIMENTAL RESULTS

To implement our algorithm we have used 16 bit TurboC compiler, Windows Xp, 400 MHz processor. For various input graph the solution of vertex cover problem and set cover problem are given in Table-1. The bolded vertexes are selected according to optimal algorithm. The dashed lines are incident edges to the selected vertexes. In the approximate algorithm of vertex cover problem, in each step it is required to add one or more edges, where in the optimal algorithm it is required to add only one vertex at a time which takes less memory than storing edges. The solution of vertex cover problem according to the approximate algorithm is {b, c, d, e, f, g}, considering the graph given in Fig. 1. The solution according to our optimal algorithm is {b, d, e} for the same graph. This solution almost 50 % better than the approximate solution for the graph given in Fig. 2. The greedy algorithm for set cover problem given in section 5, produces a cover of size 4 using Figure-8 by selecting the sets S_1, S4, S5 and S_3. But our minimum set cover algorithm produces S_3, S4 and S_5. But the time complexity is almost same. Our minimum set cover algorithm is better in case of solution. Using the datasets in Table-1, we have presented the comparison analysis in Fig. 12. In each dataset of Fig. 12, the only first bar graph (squared) represents the input vertex whilst the second and third graphs represent the output (solution) vertex using best known algorithm and our optimal algorithm. Experimental results and comparison analysis (given in Fig. 12) represent that our proposed optimal algorithms outperforms than the existing well known approximation algorithms.

8. CONCLUSION

The Optimal Vertex Cover problem is one of the fundamental NP-hard problems in the combinatorial optimization and set covering problem is also NP hard problem. As it cannot be solved exactly in polynomial time, unless P = NP, approaches have concentrated on the design of efficient approximation algorithms. Main contribution of this paper, we have presented two new algorithms for vertex cover and set cover problem that provide better solution compared to existing well known approximate algorithm. This optimal algorithm has almost the same time complexity as of the existing algorithm but with respect to the solution our proposed algorithm outperforms. The optimal vertex cover algorithm takes less memory space compared to the existing approximate vertex cover algorithm.

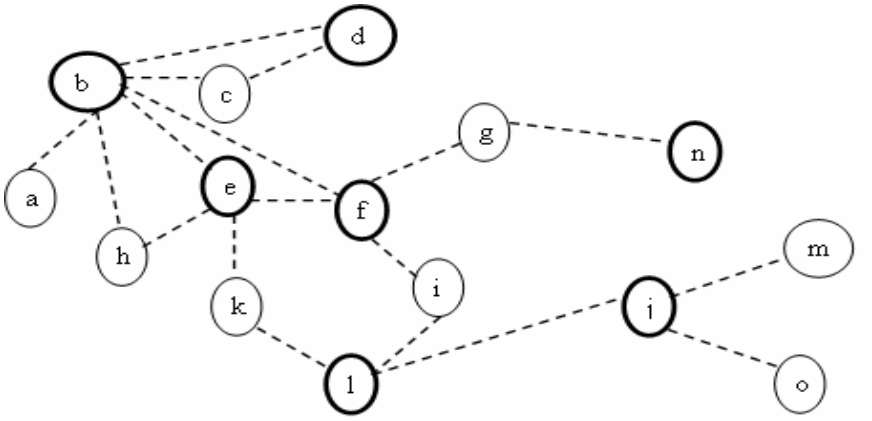

Fig. 10: Example of undirected graph, bold vertex is selected according to optimal algorithm.

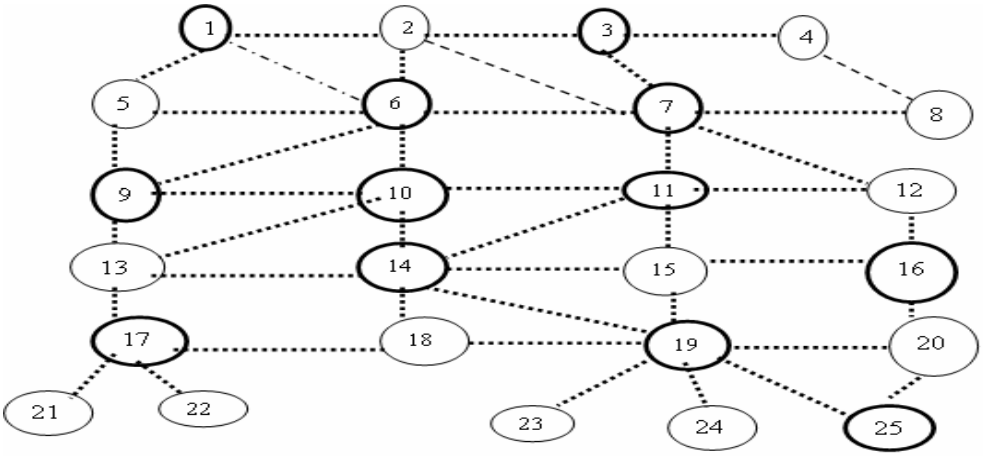

Fig. 11: Complex Undirected graph.

Table 1: Comparison Analysis using different Algorithms**.**

Dataset	Method	Problem Name	No_Of_Vertex / No_Of_Set	Output	Solution Set
Fig. 10	Existing Algorithm [2]	V-Cover	15	10	{a, b, c, h, e, f, g, I, l, n}
Fig. 10	Our Optimal Algorithm	V-Cover	15	7	{b, f, j, e, l, n, d}
Fig. 11	Existing Algorithm [2]	V-Cover	25	16	{1,3, 5, 7, 9, 11, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20}
Fig. 11	Our Optimal Algorithm	V-Cover	25	12	{19, 14, 6, 7, 17, 9, 10,11,1, 3, 16, 25}
Fig. 2	Existing Algorithm[2]	V-Cover	7	6	{b, c, d, e, f, g}
Fig. 2	Our Optimal Algorithm	V-Cover	7	3	{b, d, e}
Fig. 8	Greedy Algorithm[2]	S-Cover	6	4	{ S_1, S_3, S_4,, S_5}.
Fig. 8	Our Set cover Algorithm	S-Cover	6	3	{ S_3, S_4,, S_5}.

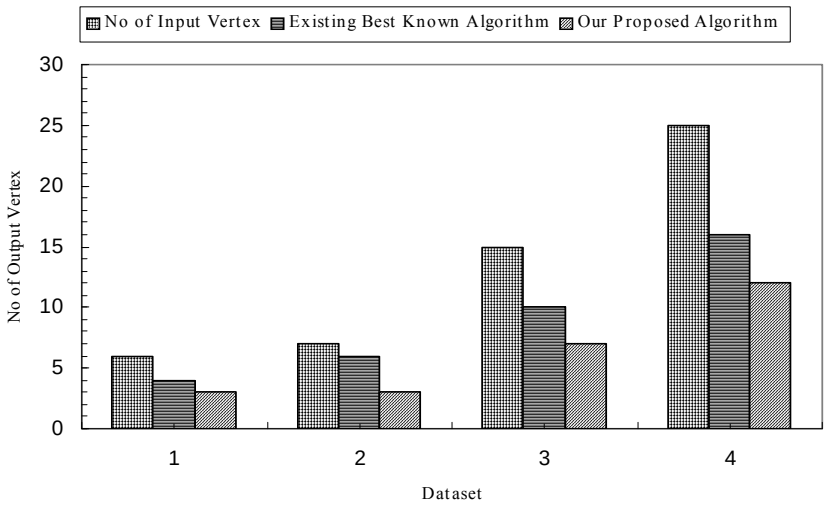

Fig. 12: Comparison analysis between existing algorithm and our proposed algorithm.

REFERENCES

[1] http://pages.cs.wisc.edu/~shuchi/courses/787-F07/scribe-notes/lecture02.pdf, Embellishment on Greedy algorithm for set cover problem.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms," MIT, September 2001.

[3] M. Chlebíka and J. Chlebíkováb, "Crown reductions for the Minimum Weighted Vertex Cover problem," Elsevier Journal, vol. 2:26, pp. 1-21, 2007.

[4] R. Duh and M. Furer, "Approximation of k-set cover by semi-local optimization," in (ACM STOC)Proceedings of the twenty-ninth annual ACM symposium on Theory of computing Texas, United States, 1997.

[5] U. Feige, "A Threshold of ln n for approximating set cover," in ACM STOC, Pennsylvania, USA, 1998, pp. 134-318.

[6] ftp://ftp.cs.buffalo.edu/pub/tech-reports/98-06.ps, "Approximation Algorithms For Set Cover And Related Problems."

[7] G. R. Quan, B. R. Hong, F. Ye, and S. J. Ren, "A heuristic function algorithm for minimum set-covering problem," Journal of Software, vol. 9:2, pp. 150-160, 1998.

[8] S. Tsukiyama, M. Ide, H. Ariyoshi, and I.Shirakawa, "A New Algorithm for Generating All the Maximum Independent Sets," SIAM Journal, vol. 6:3, pp. 505-517, 1977.

[9] P. Slavy, "A tight analysis of the greedy algorithm for set cover, " Journal of Algorithms, vol. 25:2, pp. 237-254, 1997.

[10] R. E. Tarjan, "Depth-First Search and Linear Graph Algorithm," SIAM Journal, vol. 1, pp. 146-169, 1972.

[11] Y. Wang, E. Feng, and R. Wang, "The Construction of Minimal Set-covering Model for TagSNP Selection Problem and Heuristic Function Algorithm," IEEE Journal, pp. 252-256, 2007.

[12] www.cas.mcmaster.ca / ~gk / papers / vc.pdf, "A better approximation ratio for the Vertex Cover problem."

[13] www.cs.iastate.edu / ~cs511 / handout08 / Approx_VC.pdf, "Approximation Algorithms for Weighted Vertex Cover."

[14] www.cs.iastate.edu / ~cs511 / handout08 / Set_cover.pdf, "An Approximation Algorithm for Set Cover."

[15] www.inkinghub.elsevier.com / retrieve / pii / S0020019009000696, "A simple local 3-approximation algorithm for vertex cover."

[16] F. Delbot and C. Laforest, "A better list heuristic for vertex cover," Elsevier Journal (Information Processing Letters), vol. 107, pp. 125-127, 2008.

[17] V. Polishchuk and J. Suomela, "A simple local 3-approximation algorithm for vertex cover," Elsevier Journal, vol. 109, pp. 642-645, 2009.

[18] Y. Zhang and H. Zhu, "An Approximation Algorithm for Weighted Weak Vertex Cover Problem in Undirected Graphs," LNCS, pp. 143-150, 2004.