International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:4, No:7, 2010

An Effective Algorithm for Minimum Weighted Vertex Cover problem

S. Balaji, V. Swaminathan and K. Kannan

Abstract—The Minimum Weighted Vertex Cover (MWVC) problem is a classic graph optimization NP - complete problem. Given an undirected graph G = (V, E) and weighting function defined on the vertex set, the minimum weighted vertex cover problem is to find a vertex set S⊆V whose total weight is minimum subject to every edge of G has at least one end point in S. In this paper an effective algorithm, called Support Ratio Algorithm (SRA), is designed to find the minimum weighted vertex cover of a graph. Computational experiments are designed and conducted to study the performance of our proposed algorithm. Extensive simulation results show that the SRA can yield better solutions than other existing algorithms found in the literature for solving the minimum vertex cover problem.

Keywords—weighted vertex cover, vertex support, approximation algorithms, NP-complete problem.

I. Introduction

The classical minimum weighted vertex cover problem involves graph theory and finite combinatorics and is categorized under the class of NP-complete problems[6] in terms of its computational complexity. In 1972, in a landmark paper Karp has shown that the vertex cover problem is NP-complete[13] meaning that it is exceedingly unlikely that to find an algorithm with polynomial worst-case running time. The minimum vertex cover problem remains NP - complete even for certain restricted graphs, for example, the bounded degree graphs[7]. Minimum weighted vertex cover problem (MWVC) 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 weighted vertex cover. Examples of such areas where the minimum weighted vertex cover problem occurs in real world applications are communications, particularly in wireless telecommunications, civil, electrical engineering, circuit design, network flow.

Due to computational intractability of the MWVC problem, many researchers have instead focused their attention on the design of approximation algorithm for delivering quality solutions in a reasonable time. Pitt[17] gave a randomized algorithm which randomly selects an end vertex of an arbitrary edge with a probability inversely proportional to its weight. For a comprehensive survey on the analysis of approximation algorithms for MWVC, the reader is referred to Monien and Speckenmeyer[18], Motwani[15], Hastad[11], Shyu, Yin and Lin[19] and Likas and Stafylopatis[14]. The first fixed parameter tractable algorithm for k-vertex cover problem was done by Fellows[5]. Recently, Dehne et al[4] have reported that they used fixed parameter tractable algorithm to solve the minimum vertex cover problem on coarse-grained parallel machines successfully. Neidermeier and Rossmanith[16] presented an efficient fixed parameter algorithm for the minimum weighted vertex cover problem. In this paper for efficiently solving MWVC problems, an effective algorithm called Support Ratio Algorithm (SRA) is proposed. The proposed algorithm designed with the term called support of vertices, which involves the sum of the degrees of adjacency vertices, ratio between weight and product of support and degree of vertices, to get a near smallest weighted vertex cover of the graph. Its effectiveness is shown by conducting extensive computational experiments on a large number of random graphs. The simulation results show that the SRA can find the optimum solution.

S. Balaji, Corresponding Author, Department of Mathematics, SASTRA University, Thanjavur, India. e-mail: balaji_maths@yahoo.com.
V. Swaminathan, Ramanujan Research centre, Saraswathi Narayanan College, Madurai, India. e-mail: sulanesri@yahoo.com.
K. Kannan, Department of Mathematics, SASTRA University, Thanjavur, India. e-mail: kkannan@maths.sastra.edu.

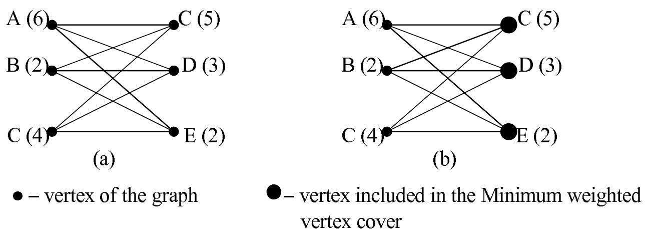

Fig. 1. (a) Graph G(V,E) (b) G(V,E) with MWVC

The paper is organized as follows. Section II briefly describes the minimum weighted vertex cover problem and its theoretical background. Section III outlines the SRA. In Section IV graph model used in the experiments is briefly described. Section V provides experiments done and their results. Section VI summarizes and concludes the paper.

II. Minimum Weighted Vertex Cover Problem

Let G = (V, E) be an undirected graph with a weight function $\omega : V \rightarrow$R associated with each vertex of v∈V, a set S⊆V is a minimum weighted vertex cover of G if (i) for every edge (u, v)∈E, either u∈S or v∈S or both u,v∈S and (ii) among all covers of E, S has the smallest weight, i.e., $\Sigma_{v\in V}\omega(v)$ is minimum. A vertex cover is minimal or optimal if it has a minimum size, i.e., if there is no vertex cover set having fewer vertices. The goal of minimum weighted vertex cover problem is to find a vertex cover of minimum weight. Fig.1 briefly explains the above criteria.

Graph of the MWVC instance shown in the Fig.1(a), where the index of the vertexes are denoted by alphabets and the number in the brackets is the weight of the associated vertexes. The optimal solution $S = \{C, D, E\}$ with a total weight of 10. However there is no vertex set, covering all the edges with total weight less than 10 shown in Fig. 1(b). There are two versions of the vertex cover problem: the decision and optimization versions. In the decision version, the task is to verify for a given graph G, a weight function $\omega : V \rightarrow$R and k, weighted vertex cover asks for a vertex cover of total weight at most k but in the optimization version the task is to find a vertex cover of minimum total weight. In this paper we consider the optimization version of the minimum weighted vertex cover with the goal of obtaining optimum solution. Now the MWVC is formulated as an integer programming problem by using the following conditions: Binary variables a_{ij}(i = 1,2,3,...,n; j = 1,2,3,...,n) form the adjacency matrix of the graph G. Each variable has only two values (1 or 0) according as an edge exists or not. In other words, if an edge (v_i,v_j) is in E, then a_{ij} is 1 else a_{ij} is 0. For example, for the graph of Fig.1 has the following adjacency matrix

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

The output of the program expresses the vertex v_i is in the vertex cover or not. v_i=1 if it is in the vertex cover otherwise v_i=0. Thus the total weight in the vertex cover can be expressed by Z = $\Sigma w_i v_i,\ 1 \leq i \leq n$. At least one vertex of the edge must be included in the vertex cover, so we have the constrained condition of the minimum vertex cover can be written as $v_i + v_j \geq$1. Thus the problem can be mathematically transformed into the following optimization problem as

Min Z = $\Sigma w_i v_i$
Subject to
$v_i + v_j \geq 1\ \forall (v_i, v_j) \in$E
$v_i \in \{0, 1\}\ \forall v_i \in$V

To illustrate the minimum vertex cover problem, consider the problem of placing guards with associated costs of guards [20] in a museum where corridors in the museum correspond to edges and task is to place a minimum number of guards with paying minimum total cost so that there is at least one guard at the end of each corridor. Fig.1 depicts the problem in brief.

Minimum weighted vertex cover problem is a special case of set cover problem[3] which takes as input an arbitrary collection of subsets $S = \{S_1, S_2, ..., S_n\}$ of the universal set V, and the task is to find a smallest subsets from S whose union is V. The minimum weighted vertex cover problem is also closely related to many other hard graph problems and so it is of interest to the researchers in the field of design of optimization and approximation algorithms. For instance the independent set problem[2][13][7] is similar to the minimum vertex cover problem because a minimum vertex cover defines a maximum independent set and vice versa. Another interesting problem that is closely related to the minimum vertex cover is the edge cover which seeks the smallest set of edges such that each vertex is included in one of the edges.

III. Terminologies, Algorithm and Computational complexity

Neighborhood of a vertex: Let G = (V, E), V is a vertex set and E is an edge set, be an undirected graph and let $|V|$ = n and $|E|$ = m. Then for each v∈V, the neighborhood of v is defined by $N(v) = \{u \in V/u\ is\ adjacent\ to\ v\}$ and N[v] = v ∪ N(v).

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

Support of a vertex: The support of a vertex v∈V is defined by the sum of the degree of the vertices which are adjacent to v, i.e., support(v) = s(v) = $\Sigma_{u \in N(v)} d_G(u)$.

A. *Support Ratio Algorithm (SRA - Proposed)*

The following algorithm is designed to find the general minimum weighted vertex cover of a graph G. Adjacency matrix of the given graph G of n vertices and m edges and the weights of the each vertexes are given as the input of the program. The degree d(v) and support s(v) of the each vertex v∈V are calculated. Support of the vertex calculated by the relation $\Sigma_{u \in N(v)} d_G(u)$. Moreover the ratio r(v) of each v∈V calculated by the relation $r(v) = \frac{s(v)d(v)}{w(v)}$. Add the vertex which has the maximum value of the ratio r(v) into the vertex cover V_c . If one or more vertices have equal maximum value of the support, in this case if ($s(v_i) \geq s(v_j)$), add the vertex v_i into the vertex cover V_c otherwise add v_j into V_c. Update the adjacency matrix by deleting N[v], v∈V_c, from the given graph G. Proceed the above process until the edge set E has no edges. i.e., up to $a_{ij} \neq 0\ \forall i, j$. The pseudo-code of the proposed algorithm is given below.

```
Input: G (V, E)
Output: Z = $\Sigma_{v_i \in V_c} w_i v_i$
while E $\neq \phi$ do
step 1:
for i←1 to n
for j ←1 to n
$d(v_i) = \Sigma_j a_{ij}$
step 2:
for i←1 to n
for j ←1 to n
$s(v_i) = \Sigma_{v_j \in N(v_i)} d_G(v_j)$
step 3:
for i←1 to n
$r(v_i) = \frac{s(v_i)d(v_i)}{w(v_i)}$
step 4:
add the vertex $v_i$, having the maximum value of r($v_i$),
into the vertex cover $V_c$
$V_c \leftarrow V_c \cup v_i$
```

```
step 5:
delete N[v],v∈ $V_c$, from G;
end while.
for i←1 to n
if($v_i \in V_c$)
$v_i = 1$
else
$v_i = 0$
end.
```

in step 4 if two or more vertices have equal maximum value of r(v), in such case, use the modified version of step 4 as

```
step 4a:
if (one or more vertices have equal maximum value
of r(v))then
$V_c \leftarrow V_c \cup v_i$   if($s(v_i) \geq s(v_j)$)
else
$V_c \leftarrow V_c \cup v_j$
go to step 5.
```

B. Computational Complexity

The worst case complexity of finding the solution of the minimum weighted vertex cover problem using SRA can be obtained as follows: Assume that there are n vertices and m edges, in the given algorithm calculation of degree of vertices in step 1, support of vertices in step 2 and the ratio r(v) of vertices in step 3 requires O(n^2),O(n^2) and O(n) running time respectively. To pick the vertex which has the maximum value of the ratio in step 4 requires O(n) running time. The procedure of the algorithm goes up to m steps (worst case). So the overall running time of the procedure of SRA can be deduced as follows: m(O(n^2)+ O(n^2)+ O(n) + O(n)) = O($mn^2 + mn^2 + mn + mn$) = O(mn^2).

IV. Graph Models

This section outlines the graph models used to assess the effectiveness of the proposed algorithm in previous section. The graph models used are (i) G(n, p) graphs[1] and (ii) G(n, m) graphs[1][21]. The models are standard random graph models from the graph theory and all the graphs are undirected.

1) G(n, p) Model: The G(n, p) model is also called Erdos Renyi random graph model[1], consists of graphs of n vertices for which the probability of an edge between any pair of nodes is given by a constant $p > 0$. To ensure that graphs are almost always connected, p is chosen so that $p >> \frac{log(n)}{n}$. To generate a G(n, p) graph we start with an empty graph. Then we iterate through all pairs of nodes and connect each of these pairs with probability p.

2) Algorithm to generate (G, n, p)graphs: The pseudo code for generating G (n, p) graphs as follows

```
initialize graph G(V, E)
for i ← 1 to n
for j← i+1 to n
add edge (i, j) to E with probability p
return (G).
```

The expected number of edges of G(n, p) graph is $pn(n-1)/2$ and expected degree is np. Graphs are generated for different p and n values.

3) G(n, m) Model: The G(n, m) model consists of all graphs with n vertices and m edges. The number of vertices n and the number of edges m are related by m = nc, where c>0 is constant. To generate a random G(n, m) graph, we start with a graph with no edges. Then, cn edges are generated randomly using uniform distribution over all possible graphs with cn edges. Each node is thus expected to connect to 2c other nodes on average. The pseudo-code for the random graph generation is shown in the following algorithm.

4) Algorithm to generate (G, n, c)graphs: The pseudo code for generating G (n, m) graphs as follows

```
initialize graph G(V, E)
$m \leftarrow n * c$
for i ← 1 to m
repeat
e ← random edge
until e not present in E
E ← E ∪ $\{e\}$
return (G).
```

V. Experimental results and analysis

All the procedures of SRA have been coded in C++ language. The experiments were carried out on an Intel Pentium Core2 Duo 1.6 GHz CPU and 1 GB of RAM. The effectiveness of the SRA heuristic was evaluated using 60 instances. These instances are divided into 3 sets as shown in the TABLE I. Simulations are carried out on three types of graphs: the randomly generated small size, moderate and large scale graphs for the minimum weighted vertex cover problem.

TABLE I
MWVC Instances

Problem set	No. of Instances	Range of Weights	Graph Model	Optimal Solution
1	20	[1,40]	G(n, p)	Known
2	20	[1, d(i)^2]	G(n, m)	Unknown
3	20	[1, 100]	G(n, m)	Unknown

A. Experiment 1

We first tested the SRA on 20 random graphs generated based on the concept explained in Section IV-1. The weight w(i) on vertex i was randomly selected in the range [1, 40], 1≤ i ≤ n. The result we recorded for each test graph and their information are shown in the TABLE II. These results have been compared with dual-LP (D-LP)[12] method, its effectiveness to solve the minimum weighted vertex cover problem shown in [10]. From the TABLE II, we can see that the SRA approach delivers the optimal solutions to the most of the MWVC test instances and the quality of the solution of the proposed algorithm is much better that of Dual-LP method.

B. Experiment 2

To test the performance of SRA approach, further we have compared with two more heuristics WtRAND-(WR)[17] and Tabu Search(TS)[8][9] for the MWVC. Here we generated the

TABLE II
SIMULATION RESULTS FOR THE 1 SET OF INSTANCES

Graph			D-LP	SRA	Opt.
V	E	p	Algorithm	(Proposed)	
20	28	0.15	520.4	500.3	500.3
25	45	0.15	649.2	625.6	625.6
30	65	0.15	779.7	750.3	750.3
40	117	0.15	995.6	960.8	960.8
45	99	0.1	976.2	945.2	945.2
45	148	0.15	1068.8	1035.1	1035.1
50	123	0.1	1093.2	1050.4	1050.4
50	184	0.15	1178.9	1150.7	1150.7
55	148	0.1	1125.4	1100.2	1100.2
55	223	0.15	1187.2	1155.8	1155.8
60	177	0.1	1239.6	1202.5	1200.1
60	265	0.15	1300.4	1263.7	1260.8
65	208	0.1	1341.2	1300.9	1300.9
65	312	0.15	1398.4	1365.5	1365.5
70	241	0.1	1379	1330.8	1330.8
70	362	0.15	1452.5	1400.5	1400.5
75	277	0.1	1461.3	1429.4	1425.9
75	416	0.15	1558.8	1503.2	1500.2
80	316	0.1	1572.2	1521	1520.3
80	474	0.15	1640.1	1602.1	1600.4

TABLE III
SIMULATION RESULTS FOR 2ND SET OF INSTANCES

Graph			D-LP	WR	TS	SRA
Label	V	E	(a)	(b)	(c)	(d)
RG1	100	100	173.3	171.6	170.2	169.2
RG2		200	629.7	631.9	614.3	607.4
RG3		300	1880.3	1844.7	1850	1807.6
RG4		400	2761.6	2768.8	2747.5	2663.3
RG5	150	150	457.9	454.3	452.6	446
RG6		300	2005.2	1993.7	1997.2	1938.1
RG7		450	2324.5	2317.9	2279.9	2238.3
RG8		600	5395.2	5374	5338.9	5168.8
RG9	200	200	486.4	484.4	482.1	468.2
RG10		400	1710	1707.8	1722.8	1643.9
RG11		600	2941.6	2925.7	2942.8	2838.6
RG12		800	4610.5	4661.1	4645.1	4437.9
RG13	250	250	445.1	438.4	436.8	428.3
RG14		500	1524.5	1513.1	1502.9	1460.7
RG15		750	3484.6	3469.1	3454.5	3314.3
RG16		1000	6375	6339.3	6327.8	6058.2
RG17	300	300	660.3	655.1	653.3	638.3
RG18		600	1642.8	1628	1630.1	1569.5
RG19		900	4782.3	4727.4	4736.9	4539.9
RG20		1200	8104	8029	7981.6	7654.7

random graphs based on the concept explained in section IV-3. The graphs are generated based on the relation m = cn, c varied from 1.25 to 2 in steps of 0.25 and for each value of c, n varied from 100 to 300 in steps of 50. For each combination of n and c, 20 random graphs are generated. The weight w(i) on vertex i be randomly distributed over the interval [1, d(i)^2], where d(i) is the degree of the vertex i, $1 \leq i \leq n$. The above discussion is possible because of larger degree (more transportation benefits) on a vertex might induce more weight (running time) on it. We summarized the tested results in TABLE III and we calculated the percentage of deviation of other heuristics compared with SRA and these results are listed in TABLE IV, in which positive values tells us that the SRA reaches the best optimum solution and negative values represents the SRA fails to reach the optimum solution than the other heuristics compared. If the values are exactly equal to zero then the SRA and the compared heuristics reaches the same optimum. Evidently there is no negative and zero values in the TABLE IV. From the results shown in the TABLES III and IV, we can see that the quality of the solution delivered by SRA are much better than the other heuristics, involved in this experiment, even though the weights on vertexes are proportional to the degrees.

TABLE IV
DEVIATION OF OTHER HEURISTICS FROM SRA FOR 2ND SET OF INSTANCES

Label	Percentage of deviation from SRA		
	[(a-d)/d]x100	[(b-d)/d]x100	[(c-d)/d]x100
RG1	2.45	1.45	0.75
RG2	3.67	2.86	1.13
RG3	4.02	2.05	2.35
RG4	3.69	3.96	3.16
RG5	2.67	1.87	1.47
RG6	3.46	2.87	3.05
RG7	3.85	3.56	1.86
RG8	4.38	3.97	3.29
RG9	3.89	3.46	2.96
RG10	4.02	3.89	4.80
RG11	3.63	3.07	3.67
RG12	3.89	5.03	4.67
RG13	3.93	2.35	1.98
RG14	4.37	3.59	2.89
RG15	5.14	4.67	4.23
RG16	5.23	4.64	4.45
RG17	3.45	2.63	2.35
RG18	4.67	3.73	3.86
RG19	5.34	4.13	4.34
RG20	5.87	4.89	4.27
Average	4.08	3.44	3.08

C. *Experiment 3*

In this experiment the parameter set opted like small-large scale problems, that is V varied from 50 to 1000. The weight w(i) on vertex i was also randomly drawn from the interval [1, 100]. Here we used the G(n, m) graph model to generate the random graphs. All of the heuristics implemented in experiment 1 and 2 were examined in this experiment. For most of the test instances the optimal solutions are unknown, we obtained the relative performance of the other heuristics with the SRA by calculating the percentage of deviation of other heuristics from the SRA. These results are shown in the Fig. 2 where the major axis represents the 20 test instances and for each test instances error rate of other heuristics with SRA were plotted as points and for each algorithm their points

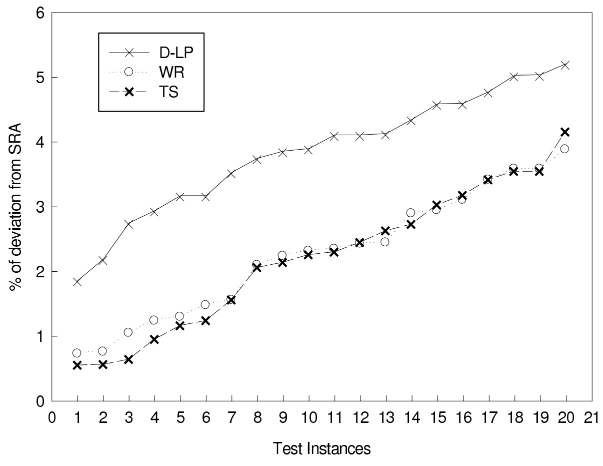

Fig. 2. Error rate (in percentage) of other heuristics with SRA in 3rd set of test instances

are linked by a line. With that figure we show that, for the set instances we used, the SRA produced better solutions than other heuristics compared and the other heuristics get higher deviation from SRA when the size of the problem increases.

VI. Conclusion

A new SRA for minimum weighted vertex cover problem has been proposed and its effectiveness has been shown by simulation experiments. The terminology support of a vertex introduced in the new model, with that, the new model can find the minimum weighted vertex cover effectively.The simulation results show that the new SRA can yield better solutions than D-LP, WR and TS heuristics for random graphs. At the same time, our approach gives best solutions for large scale problems also. The proposed algorithm has led to give near optimal solutions for most of the test instances where we know the optimal solutions. Our approach is heuristic with O(mn^2) complexity and maximum number of n iterations to solve the vertex cover problem. Furthermore, another important attractiveness of this heuristic is its outstanding performance for solving MWVC problems.

References

[1] Bollobas. B : Random graphs (2nd Ed.). Cambridge, UK: Cambridge University press, 2001.
[2] Berman. P and Fujito. T: On approximation properties of the independent set problem for low degree graphs, Theory of Computing Syst., vol. 32, pp. 115 - 132, 1999.
[3] Cormen. T. H, C. E. Leiserson, R. L. R., and Stein. C: Introduction to algorithms, 2nd ed., McGraw - Hill, New York , 2001.
[4] Dehne. F, et al.: Solving large FPT problems on coarse grained parallel machines, Available: http://www.scs.carleton.ca/fpt/papers/index.htm.
[5] Fellows. M. R: On the complexity of vertex cover problems, Technical report, Computer science department, University of New Mexico, 1988.
[6] Garey. M. R, Johnson. D. S: Computers and Intractability: A Guide to the theory NP - completeness. San Francisco: Freeman ,1979.
[7] Garey. M. R, Johnson. D. S, and Stock Meyer. L: Some simplified NP - complete graph problems, Theoretical computer science, Vol.1563, pp. 561 - 570, 1999.
[8] Glover. F: Tabu Search - Part I, ORSA journal of computing, vol. 1, No.3, (1989), pp. 190 - 206.
[9] Glover. F: Tabu search: A Tutorial, Interface 20, pp. 74 - 94, 1990.
[10] Gomes. F. C, Meneses. C. N, Pardalos. P. M and Viana. G. V. R: Experimental analysis of approximation algorithms for the vertex cover and set covering problems, Journal of computers and Operations Research, vol. 33, pp. 3520 - 3534, 2006.
[11] Hastad. J: Some Optimal Inapproximability Results., Journal of the ACM, vol. 48, No.4, pp. 798 - 859, 2001.
[12] Hochbaum. D. S: Approximation algorithm for the set covering and vertex cover problems, SIAM Journal on computing, 11(3), 555 - 6, 1982.
[13] Karp. R. M: Reducibility among combinatorial problems, Plenum Press, New York, pp. 85 - 103, 1972.
[14] Likas, A and Stafylopatis, A: A parallel algorithm for the minimum weighted vertex cover problem, Information Processing Letters, vol. 53, pp. 229 - 234, 1995.
[15] Motwani. R: Lecture Notes on Approximation Algorithms, Technical Report, STAN-CS-92-1435, Department of Computer Science, Stanford University, 1992.
[16] Neidermeier. R and Rossmanith. P: On efficient fixed-parameter algorithms for weighted vertex cover, Journal of Algorithms, vol. 47, pp. 63 - 77, 2003.
[17] Pitt. L: A Simple Probabilistic Approximation Algorithm for Vertex Cover, Technical Report, YaleU/DCS/TR-404, Department of Computer Science, Yale University, 1985.
[18] Monien. B and Speckenmeyer. E: Ramsey numbers and an approximation algorithm for the vertex cover problems, Acta Informatica, vol. 22, pp. 115 - 123, 1985.
[19] Shyu. S.J, Yin. P.Y and Lin. B.M.T: An ant colony optimization algorithm for the minimum weight vertex cover problem, Annals of Operations Research, Vol. 131, pp. 283 - 304, 2004.
[20] Weight. M and Hartmann. A. K: The number of guards needed by a museum - a phase transition in vertex covering of random graphs., Phys - Rev. Lett., 84, 6118, 2000b.
[21] Weight. M and Hartmann. A. K.: Minimal vertex covers on finite-connectivity random graphs - A hard-sphere lattice-gas picture, Phys. Rev. E, 63, 056127.