Approximating Maximum Weighted Independent Set Using Vertex Support

S. Balaji, V. Swaminathan and K. Kannan

Abstract—The Maximum Weighted Independent Set (MWIS) problem is a classic graph optimization NP-hard problem. Given an undirected graph $G = (V, E)$ and weighting function defined on the vertex set, the MWIS problem is to find a vertex set $S \subseteq V$ whose total weight is maximum subject to no two vertices in $S$ are adjacent. This paper presents a novel approach to approximate the MWIS of a graph using minimum weighted vertex cover of the graph. Computational experiments are designed and conducted to study the performance of our proposed algorithm. Extensive simulation results show that the proposed algorithm can yield better solutions than other existing algorithms found in the literature for solving the MWIS.

Keywords—weighted independent set, vertex cover, vertex support, heuristic, NP - hard problem.

I. INTRODUCTION

In graph theory an independent set of a graph is a subset of vertices in which no two vertices are adjacent (i.e., connected by an edge) and the maximum independent set problem (MIS) calls for finding the independent set of maximum cardinality. The MIS is a classic one in computer science and in graph theory and it has many important applications, including combinatorial auctions[7], graph coloring[13], coding theory[9], geometric tiling, fault diagnosis, pattern recognition, molecular biology[11], and more recently its application in bioinformatics[16] have become important. The Maximum Weighted Independent Set (MWIS) is a generalization of MIS problem, in which vertices have positive weight and one has to find an independent set of maximum weight and both MIS and MWIS problems are known to be NP-hard[12]. Hence simple algorithms which yield acceptable solutions sufficiently fast are quite important for such related practical problems.

Pardalos and Xue[15] recently published a review with 260 references. This problem is computationally intractable even to approximate with certain absolute performance bounds[6, 10]. Very few numbers of algorithms has been proposed for the MWIS problem. Babel[1] proposed an efficient branch and bound procedure. Oстергарт[14] also developed a fast branch and bound based exact method. Bonze et al.[4, 5] proposed a method based on replicator dynamics. It uses the continuous formulation of maximum weight clique problem. Recently, Pullan[18] proposed a local search procedure for maximum independent set problem, which was supported by computational experiments.

In this paper for efficiently solving MWIS problems, an effective algorithm called Support Ratio Algorithm (SRA) is proposed. This algorithm effectively finds MWIS of a graph $G$ by finding the Minimum Weighted Vertex Cover (MWVC) of the graph $G$. 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. It’s effectiveness for finding the MWVC of the graph shown in[2]. We compared our algorithm with the other existing algorithm namely[1, 5, 14, 18]. The experimental result shows that our algorithm is very fast and yields better solutions than the compared algorithms for many random graphs and DIMACS benchmark graphs.

The paper is organized as follows. Section II briefly describes the maximum weighted independent set (MWIS) problem and the minimum weighted vertex cover problem (MWVC) 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. MAXIMUM WEIGHTED INDEPENDENT SET AND MINIMUM WEIGHTED VERTEX COVER

Let $G = (V, E)$ be an arbitrary undirected graph, where $V = \{1, 2, ..., n\}$ is the set of vertices and $E \subseteq V \times V$ (not in ordered pairs) is the set of edges. Two distinct vertices $u$ and $v$ are called adjacent if they are connected by an edge; an independent set $S$ of $\mathbb{G}$ is a subset of $V$ whose elements are pairwise non-adjacent. The maximum independent set (MIS) problem seeks to find an independent set with large number of vertices. The size of the maximum independent set of $\mathbb{G}$ is the stability number of $\mathbb{G}$ and is denoted by $\alpha$. MWIS problem is a generalization of MIS in which vertices have positive weight i.e., a weight function $\omega : V \rightarrow \mathbb{R}$ associated with each vertex of $v \in V$ and one has to find the independent set with maximum weight.

A vertex cover for $G$ is a subset $V_c$ of $V$ such that for each edge $(u, v) \in E$, at least one of $u$ or $v$ or both belongs to $V_c$. The minimum vertex cover (MVC) problem consists of identifying the vertex cover $V_c$ of $G$ which has minimum cardinality and the size of the minimum vertex cover of $G$ is denoted by $\beta$. The MIS and MVC problems are related in that the maximum independent set $S$ of $G$ contains all those vertices that are
not in the minimum vertex cover of G. i.e. $S = V - V_c$ and
$\alpha + \beta = n$. The relationship identified above for MIS and
MVC problems also hold for MWIS and MWVC instances and, in
addition, if $W_T$ is the total weight of the vertices in G and
$W_C$. $W_S$ denotes the total weights in minimum weighted
vertex cover, the maximum weighted independent set in G
then $W_T = W_C + W_S$. Fig.1 briefly explains the above
criteria. Graph of MWIS and MWVC instances shown in the
Fig. 1, where the index of the vertices are denoted by alphabets
and the number in the brackets is the weight of the associated
vertices. The optimal solution for MWIS is $S = \{A, B, E, F\}$
with a total weight of 18 and MWVC is $V_c = \{C, D\}$ with
total weight of 5. However there is no vertex set in G with
pairwise non adjacent vertices and total weight greater than 18
and no vertex set in G covering all the edges with total weight
less than 5. Moreover we can see clearly that $W_T = W_C + W_S$.

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 minimum weighted vertex cover
problem can be formulated as an integer programming prob-
lem 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

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

The output of the program expresses the vertex $v_i$ is in
the MWIS or not. $v_i=1$ if it is in the MWIS otherwise $v_i=0$.
Thus the total weight in the MWIS can be expressed by $Z
= \sum_{v_i} a_{ij}, 1 \leq i \leq n$. Any one or none of the vertex
of the edge $(v_i, v_j)$ is included in the independent set, so we
have the constrained condition of the minimum vertex cover
be written as $v_i + v_j \leq 1$. Thus the problem can be
mathematically transformed into the following optimization
problem as

$$\text{Max } Z = \sum_{v_i} a_{ij}$$
Subject to
$$v_i + v_j \leq 1 \ \forall (v_i, v_j) \in E$$
$$v_i \in \{0, 1\} \ \forall v_i \in V$$

Thus we find the MWIS of the graph G by finding MWVC
of the graph G. i.e., if the set $V_c$ contains the vertices of
MWVC of the graph then the set $S$ of all vertices of MWIS
of the graph G can be extracted from MWVC by $S = V - V_c$.

III. TERMINOLOGIES AND PROPOSED ALGORITHM

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 $v \in V$, the neighborhood of $v$
swas defined by $N(v) = \{u \in V : u \text{ is adjacent to } v\}$ and $N[v] = v \cup N(v)$.

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

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

A. Support Ratio Algorithm (SRA - Proposed)

The following algorithm is designed to find the general
maximum weighted independent set of a graph G. Adjacency
matrix $(a_{ij})$ 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 each
vertex $v \in V$ are calculated. Support of the vertex calculated
by the relation $\sum_{u \in N(v)} d_G(u)$. Moreover the ratio $r(v)$ of
each $v \in V$ calculated by the relation $r(v) = \frac{s(v)}{d(v)}$. 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 of G by putting zero
in to the row and column entries of the corresponding vertex
$v \in V_c$. Proceed the above process until the edge set E has
no edges. I.e., $a_{ij} \neq 0 \ \forall i, j$. The pseudo-code of
the proposed algorithm is given below.

Input: G (V, E)
Output: Max. Weighted Independent Set $S(G) = V - V_c$
and $Z = \sum_{v_i \in S} d_i e_i$
while $E \neq \phi$
do
step 1:
for $i$ in 1 to $n$
for $j$ in 1 to $n$
d$\omega_{ij} = \sum_{a_{ij}}
step 2:
for $i$ in 1 to $n$
for $j$ in 1 to $n$
$s(v_i) = \sum_{v_j \in N(v_i)} \delta_G(v_j)$

**Step 3:**

for $i \leftarrow 1$ to $n$

$r(t_i) = \frac{s(v_i)}{w(t_i)}$

**Step 4:**

$max = r(t_1)$;

$k = 1$;

select the vertex which has the maximum value of $r(v)$ in to $V_c$

for $i \leftarrow 2$ to $n$

if $(max < r(v_i))$

$max = r(t_i)$;

$t = i$;

$V_c \leftarrow V_c \cup v_i$

end if

if multiple vertices have equal maximum value of $s(v)$

then follow step 4a

**Step 4a:**

if $(max = r(v_i) \& \& s(v_i-k) <= s(v_i))$

$max = r(t_i)$;

$t = i-k$;

$V_c \leftarrow V_c \cup v_i$

end if

if $(max = r(v_i) \& \& s(v_i-k) > s(v_i))$

$max = r(t_{i-k})$;

$t = i-k$;

$V_c \leftarrow V_c \cup v_i$

end if

$k = k+1$;

end for

**Step 5:**

for $i \leftarrow 1$ to $n$

$(a_{ti}) = 0$;

$(a_{di}) = 0$;

end for

end while.

**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[3] and (ii) G(n, m) graphs[3][19]. 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[3], consists of graphs of n vertices for which the probability of an edge between any pair of nodes is given by a constant $p \geq 0$. To ensure that graphs are almost always connected, $p$ is chosen so that $p \gg \frac{\log n}{n}$. 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 $i \leftarrow 1$ to $n$

for $j \leftarrow i+1$ to $n$

add edge (i, j) to E with probability $p$

return (G).

The expected number of edges of G(n, p) graph is $pm(n-1)/2$ 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 $m = cn$, 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)

$m \leftarrow n \times c$

for $i \leftarrow 1$ to $m$

repeat

$e \leftarrow$ random edge

until $e$ not present in E

$E \leftarrow E \cup \{e\}$

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 117 instances. These instances are divided into 3 sets as shown in the TABLE I. Simulations are carried on three types of graphs: the randomly generated small size, moderate and large scale graphs for the maximum weighted independent set problem.

<table>
<thead>
<tr>
<th>Problem set</th>
<th>No. of Instances</th>
<th>Range of Weights</th>
<th>Graph Model</th>
<th>Optimal Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>44</td>
<td>[1, 40]</td>
<td>G(n, p)</td>
<td>Known</td>
</tr>
<tr>
<td>2</td>
<td>53</td>
<td>[1, 10]</td>
<td>DIMACS</td>
<td>Unknown</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>[1, 100]</td>
<td>G(n, m)</td>
<td>Unknown</td>
</tr>
</tbody>
</table>

**A. Experiment I**

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], $1 \leq i \leq n$. The result we recorded for each test graph and their information are shown in the TABLE II. $W_S$ is the total weight of the independent set found by corresponding algorithm and $\alpha(G)$ represents the cardinality of the maximum weighted
independent set. The results are compared with Ostergard[14] method. From the TABLE II, we can see that the SRA approach delivers the optimal solutions to the most of the MWIS test instances and the quality of the solution of the proposed algorithm is much better that of Ostergard method. We are interested to test the proposed algorithm on high density and large size (number of vertices) random graphs. For these large size random graphs we have chosen the same range of weights and the result we recorded are shown in the TABLE III. The time taken (in seconds) to find the MWIS for each of the instances are reported, in which the maximum time taken of 28 sec. (1000; 0.9) is an encouraging one and it is comparatively very less time for the graph of high density and large number of vertices.

### TABLE II
**Simulation results for 1st set of instances**

<table>
<thead>
<tr>
<th>Graph</th>
<th>Ostergard</th>
<th>SRA</th>
<th>Optimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>p</td>
<td>α_G</td>
<td>W_C</td>
</tr>
<tr>
<td>20</td>
<td>0.85</td>
<td>5</td>
<td>166.7</td>
</tr>
<tr>
<td>25</td>
<td>0.85</td>
<td>8</td>
<td>294.4</td>
</tr>
<tr>
<td>30</td>
<td>0.85</td>
<td>7</td>
<td>228.3</td>
</tr>
<tr>
<td>40</td>
<td>0.85</td>
<td>9</td>
<td>278.9</td>
</tr>
<tr>
<td>45</td>
<td>0.85</td>
<td>9</td>
<td>236.3</td>
</tr>
<tr>
<td>50</td>
<td>0.85</td>
<td>6</td>
<td>159.2</td>
</tr>
<tr>
<td>50</td>
<td>0.9</td>
<td>6</td>
<td>143.2</td>
</tr>
<tr>
<td>50</td>
<td>0.85</td>
<td>4</td>
<td>100.1</td>
</tr>
<tr>
<td>55</td>
<td>0.9</td>
<td>10</td>
<td>244.5</td>
</tr>
<tr>
<td>55</td>
<td>0.85</td>
<td>7</td>
<td>168.5</td>
</tr>
<tr>
<td>60</td>
<td>0.9</td>
<td>11</td>
<td>269.4</td>
</tr>
<tr>
<td>60</td>
<td>0.85</td>
<td>8</td>
<td>190.2</td>
</tr>
<tr>
<td>65</td>
<td>0.9</td>
<td>7</td>
<td>157</td>
</tr>
<tr>
<td>65</td>
<td>0.85</td>
<td>5</td>
<td>113.8</td>
</tr>
<tr>
<td>70</td>
<td>0.9</td>
<td>7</td>
<td>147.9</td>
</tr>
<tr>
<td>70</td>
<td>0.85</td>
<td>5</td>
<td>107.7</td>
</tr>
<tr>
<td>75</td>
<td>0.9</td>
<td>5</td>
<td>116.4</td>
</tr>
<tr>
<td>75</td>
<td>0.85</td>
<td>6</td>
<td>122.4</td>
</tr>
<tr>
<td>80</td>
<td>0.9</td>
<td>10</td>
<td>193.4</td>
</tr>
<tr>
<td>80</td>
<td>0.85</td>
<td>7</td>
<td>157.3</td>
</tr>
</tbody>
</table>

### TABLE III
**SRA performance on large size graph of 1st set**

<table>
<thead>
<tr>
<th>Graph</th>
<th>p</th>
<th>α_G(G)</th>
<th>W_C</th>
<th>α_G(G)</th>
<th>W_C</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.7</td>
<td>15</td>
<td>375</td>
<td>18</td>
<td>26</td>
</tr>
<tr>
<td>200</td>
<td>0.7</td>
<td>19</td>
<td>475</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>300</td>
<td>0.7</td>
<td>21</td>
<td>529</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>400</td>
<td>0.7</td>
<td>23</td>
<td>602</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>500</td>
<td>0.7</td>
<td>28</td>
<td>688</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>600</td>
<td>0.7</td>
<td>31</td>
<td>744</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>700</td>
<td>0.7</td>
<td>36</td>
<td>821</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>800</td>
<td>0.7</td>
<td>37</td>
<td>858</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>900</td>
<td>0.7</td>
<td>43</td>
<td>912</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>1000</td>
<td>0.7</td>
<td>47</td>
<td>977</td>
<td>12</td>
<td>12</td>
</tr>
</tbody>
</table>

### B. Experiment 2

As there are no resulted benchmark set for the maximum weighted clique, to test the performance of SRA approach, further we have tested the proposed algorithm on benchmark graphs of maximum clique, which are made available by DIMACS [8], and this suite structured from the perspective of finding maximum cliques, so we considered the benchmark graphs as $G$. These DIMACS instances were converted into weighted benchmark instances by allocating $w(i)$, for vertex $i$ ($1 \leq i \leq n$), uniformly in the interval [1 - 10]. The result we recorded are shown in TABLE IV, in which the first two columns reports the name and size of the graphs; For the MWIS obtained by SRA, the third column $W_C$ reports the total weight; the fourth column $\alpha_G$ reports the cardinality; the fifth column $A(C)$ gives the average vertex weight and sixth column delivers the the time (in sec.) taken to find the MWIS.

To check whether the SRA reaches the optimum (best solution) for these maximum weighted clique instances, we took the experimental results reported in Bomze et al 2000, Babel 1994 and Pullan 2008 and we analyzed the percentage of deviation of these heuristics with the proposed algorithm. i.e., for some of these instances of same condition, we compared the performance of SRA with other heuristics Babel[1], RD[5] and PLS[17, 18]. These results are reported in TABLE V and from the last three columns of TABLE V, we can see that positive values tells us that 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 SRA and compared heuristics reaches the same optimum. Out of 23 compared instances, for the instances c-fat500-1 and p_hat1500-3, SRA fails to reach the optimum than PLS and SRA reaches the same optimum with PLS in brock800.1, brock800.2 and p_hat1500-1 instances and with Babel in c-fat500-1 instance and for the remaining instances the SRA reaches the best solution.

In order to assess the amount of deviation of other heuristics from SRA, further we obtained the statistical quantities of last three columns values of TABLE V and obtained results are shown in TABLE VI. It shows that Babel heuristic deviated highly and PLS gets low deviation from SRA. From these results shown in TABLES V and VI, we can see that the quality of the solution delivered by SRA is much better than the other heuristics, involved in this experiment.
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. For most of the test instances the optimal solutions are unknown, we obtained the time (in sec.) taken by the SRA for finding the MWIS of the graph. These results are shown in the Fig. 2 where the major axis represents the size (in terms of number of vertices) of the 20 test instance’s and for each test instances the time taken by SRA were plotted as points and for each instances their points are linked by a line. It is clear from the Fig. 2 that the time taken by the SRA to find the optimum value of each of the MWIS instances increases steadily when the size of the problem increases and the maximum time taken is 12.31 sec. With this figure we show that the proposed algorithm took very less time to produce a maximum weighted clique for each of the test instances.

<table>
<thead>
<tr>
<th>Name</th>
<th>RD</th>
<th>Babel</th>
<th>PLS</th>
<th>% deviation from SRA</th>
</tr>
</thead>
<tbody>
<tr>
<td>brock800-1</td>
<td>136</td>
<td>156</td>
<td>163</td>
<td>16.56</td>
</tr>
<tr>
<td>brock800-2</td>
<td>142</td>
<td>157</td>
<td>163</td>
<td>12.88</td>
</tr>
<tr>
<td>brock800-3</td>
<td>141</td>
<td>148</td>
<td>162</td>
<td>14.54</td>
</tr>
<tr>
<td>brock800-4</td>
<td>136</td>
<td>147</td>
<td>161</td>
<td>20.93</td>
</tr>
<tr>
<td>c-fat500-1</td>
<td>93</td>
<td>96</td>
<td>97</td>
<td>3.13</td>
</tr>
<tr>
<td>c-fat500-2</td>
<td>181</td>
<td>181</td>
<td>168</td>
<td>2.69</td>
</tr>
<tr>
<td>c-fat500-5</td>
<td>371</td>
<td>395</td>
<td>418</td>
<td>12.29</td>
</tr>
<tr>
<td>hamming10-2</td>
<td>2674</td>
<td>2193</td>
<td>2853</td>
<td>7.98</td>
</tr>
<tr>
<td>hamming10-4</td>
<td>258</td>
<td>277</td>
<td>306</td>
<td>18.35</td>
</tr>
<tr>
<td>MANN=9</td>
<td>2643</td>
<td>2653</td>
<td>2133</td>
<td>0.56</td>
</tr>
<tr>
<td>san1000</td>
<td>80</td>
<td>94</td>
<td>90</td>
<td>15.78</td>
</tr>
<tr>
<td>p_hat1000-1</td>
<td>77</td>
<td>80</td>
<td>82</td>
<td>11.49</td>
</tr>
<tr>
<td>p_hat1000-2</td>
<td>285</td>
<td>284</td>
<td>293</td>
<td>4.04</td>
</tr>
<tr>
<td>p_hat1000-3</td>
<td>435</td>
<td>349</td>
<td>407</td>
<td>0.22</td>
</tr>
<tr>
<td>p_hat1500-1</td>
<td>74</td>
<td>87</td>
<td>91</td>
<td>18.68</td>
</tr>
<tr>
<td>p_hat1500-2</td>
<td>327</td>
<td>309</td>
<td>427</td>
<td>24.12</td>
</tr>
<tr>
<td>p_hat1500-3</td>
<td>490</td>
<td>390</td>
<td>586</td>
<td>15.95</td>
</tr>
<tr>
<td>p_hat500-1</td>
<td>56</td>
<td>72</td>
<td>75</td>
<td>28.20</td>
</tr>
<tr>
<td>p_hat500-2</td>
<td>213</td>
<td>228</td>
<td>243</td>
<td>13.76</td>
</tr>
<tr>
<td>p_hat500-3</td>
<td>307</td>
<td>294</td>
<td>338</td>
<td>10.23</td>
</tr>
<tr>
<td>p_hat700-1</td>
<td>54</td>
<td>76</td>
<td>76</td>
<td>32.5</td>
</tr>
<tr>
<td>p_hat700-2</td>
<td>275</td>
<td>272</td>
<td>283</td>
<td>4.51</td>
</tr>
<tr>
<td>p_hat700-3</td>
<td>371</td>
<td>316</td>
<td>397</td>
<td>8.39</td>
</tr>
</tbody>
</table>

### TABLE VI

**Statistical values of % of deviation**

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Min.</th>
<th>Median</th>
<th>Average</th>
<th>Max.</th>
<th>Std. Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD</td>
<td>0.23</td>
<td>12.88</td>
<td>12.95</td>
<td>32.5</td>
<td>8.58</td>
</tr>
<tr>
<td>Babel</td>
<td>0</td>
<td>7.69</td>
<td>10.85</td>
<td>33.83</td>
<td>10.09</td>
</tr>
<tr>
<td>PLS</td>
<td>-1.04</td>
<td>1.81</td>
<td>3.37</td>
<td>19.75</td>
<td>4.47</td>
</tr>
</tbody>
</table>
A new SRA for MWIS in a graph using MWVC 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 maximum weighted independent set effectively. Experimental result shows that this approach greatly reduce the execution time and in addition, the simulation results show that the new SRA can yield better solutions than Babel, Ostergard, RD and PLS heuristics found in the literature. At the same time, our approach gives best solutions for DIMACS weighted instances and also for random graphs. The proposed algorithm has led to give near optimal solutions for most of the test instances where we know the optimal solutions. Furthermore attractiveness of this heuristic is its outstanding performance for solving MWIS.

VI. CONCLUSION

A new SRA for MWIS in a graph using MWVC 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 maximum weighted independent set effectively. Experimental result shows that this approach greatly reduce the execution time and in addition, the simulation results show that the new SRA can yield better solutions than Babel, Ostergard, RD and PLS heuristics found in the literature. At the same time, our approach gives best solutions for DIMACS weighted instances and also for random graphs. The proposed algorithm has led to give near optimal solutions for most of the test instances where we know the optimal solutions. Furthermore attractiveness of this heuristic is its outstanding performance for solving MWIS.

REFERENCES