HEURISTIC ALGORITHM FOR GRAPH COLORING
BASED ON MAXIMUM INDEPENDENT SET

Hilal Al Mara’beh, Amjad Suleiman

Department of Basic Science King Saud bin Abdulaziz University for Health Sciences (KSAU-HS), Saudi Arabia

ABSTRACT: A number of heuristic algorithms have been developed for the graph coloring problem, but unfortunately, on any given instance, they may produce colorings that are very far from optimal. In this paper we investigated and introduce a three heuristics algorithm to color a graph based on a maximum independent set. The select a node with minimum and maximum degree consecutively (Min_Max) algorithm is implemented, tested and compared with select a node with minimum degree first (SNMD), select a node with maximum degree first (SNXD) in terms of CPU time and size of graph with different densities (0.1, 0.2, ..., 0.9). The result indicated that the Min_Max algorithm and SNXD is better than SNMD based on the time of first maximum independent set, running time of CPU and the number of coloring nodes.

KEYWORDS: Maximum independent set, Heuristic algorithm, adjacent nodes, Independent set.

I. INTRODUCTION

A coloring of graph G = (V, E) with vertex set V and edges set E is a mapping c: V → F, where F is a finite set. The elements of F will be referred to as the colors; so that colloquially, the definition basically implies that adjacent vertices cannot be assigned that same color; [AS99]. When a coloring exists, and |c(V)| = k, c is said to be a k-coloring of G. Notice that a k-coloring partitions V into k sets V1, ...,Vk, such that no two vertices in that same set are adjacent. The chromatic number X(G) is the smallest number of colors needed to color G, that is, the smallest k such that three exists a coloring c for G and |c(V)| = k.

The simplest heuristic methods are sequential coloring strategies, and example can be found in [GJ97]. This algorithm can easily be implemented so that its worst-case complexity is O(n^2). One of the most successful deterministic heuristic is the recursive largest first (RLF) producer due to Leighton [Lag02], which is based on the idea of coloring as much as possible with on color before introducing another. Johnson, et al. [Joh74] developed a randomize version of this method called XRRLF that improves the performance of the simple heuristic. Met heuristic such as simulated annealing [J+91], genetic algorithm [KGV83] and GRASP [GL97, Hol75], have also been applied to graph coloring, Hertz and de Werra [FR89] present result of Appling simulated annealing to graph coloring. There are many application of graph coloring such as, software engineering, assigning channels to radio station and register allocation in a computer.

Solving graph coloring problem based on the maximum independent set problem (MIS), which that of finding (MIS) is important for many applications such as computer science, engineering.[FR95].

Graph coloring is a well-known NP-hard problem [CHW87]; this is why a variety of heuristic approaches have been Min_Max to produce good colorings in a reasonable amount of time. The purpose of present work is to introduce more efficient way to find the minimum number of color needed to color any type of graph by finding number of maximum independent sets of a graph with polynomial complexity.

The next section will talk about SNMD algorithm, and solve an example by SNMD in order to show how it works. Then, section 3 discusses the SNXD algorithm and solves an example by SNXD in order to show how it works. The section 4, discusses the Min_Max algorithm in details. In section 5, we present the experimental performance of the three algorithm and some comparisons between some factors like number of nodes on a graph with independence number, and number of color required at different densities (0.1, 0.2, ..., 0.9) finally, concluding remarks are made in section 6.

II. SELECT A NODE WITH MINIMUM DEGREE FIRST (SNMD)

The process of this approach is to select the node that have the minimum degree among all nodes of a graph, add it to approximated maximum independent set, and delete the selected node and its neighbors from the graph. The process is repeated until the degree of the remaining graph become zero; [Joh74].
Procedure SNMD:
Input: G (V, E)
Output: MIS
/* Global : int max_ind_set[], char visite_node[]; 
Get_ind_set() /* function to get the independent sets 
{ While degree_node(G) <= 0 /* return zero if there are no nodes 
{ v:=select_node(G) /* find the node that has smallest degree , if there are more than one choose the first one. 
G:=G-v-neighbors(v); 
max_ind_set=max_ind_set U {v} 
} 
} 

The running time bound for SNMD is $O(n^2)$, in Fig.2 we apply this algorithm on a graph in Fig.1. 
As a result in Fig.2 the procedure is explained in the following steps: 
Step 1: choose the node that has minimum degree first and put it in max_ind_set[], as you see there is tow node has the same degree , so we choose node that has ascending order [node 2].
Step 2: delete the selected node and its neighbors from the graph.
Step 3: after deleting the graph containing tow nodes [3,5], apply algorithm on this sub graph to select another node, after applying the algorithm the Maximum Independent Set is [2,3].
Step 4: delete the Maximum Independent Set nodes from the main graph, and apply the algorithm once more to find all coloring until the degree of the remaining graph becomes zero.
When the algorithm finish, we will obtain on four colors for the entire main graph: 
1: [2,3] MIS.
2: [1].
3: [4].
4: [5].

III.SELECT A NODE WITH MAXIMUM DEGREE FIRST (SNXD)

The process of this approach is to select the node that have the maximum degree among all nodes of a graph, add it to approximated maximum independent set, and delete the selected node and its neighbors from the graph. The process is repeated until the degree of the remaining graph become zero;[GJ97].

Procedure SNXD:
Input: G(V,E)
Output: MIS
/* Global: int max_ind_set[], char visite_node[]; 
Get_ind_set() /* function to get the independent sets 
{ 
While degree_node(G) <> 0 /* return zero if there are no nodes 
{ 
v:=select_node(G) /* find the node that has largest degree , if there are more than one choose the first one. 
G:=G-v-neighbors(v); 
max_ind_set=max_ind_set U {v} 
} 
} 
The running time bound for SNXD is $O(n^2)$, in Fig.3 we apply this algorithm on a graph in Fig.1.
As a result in Fig.3 the procedure is explained in the following steps:
Step 1: choose the node that has maximum degree first and put it in max_ind_set[], as you see there is one node [1].
Step 2: delete the selected node and its neighbors from the graph.
Step 3: after deleting the graph containing three nodes zero nodes, the Maximum Independent Set is [1].
Step 4: delete the Maximum Independent Set nodes from the main graph, and apply the algorithm once more to find all coloring until the degree of the remaining graph becomes zero.
When the algorithm finish, we will obtain on three colors for all the main graph:
1: [1] MIS.
2: [4,3].
3: [2,5].

Fig.1. G=(V,E)

Fig.2. SNMD Coloring Graph
IV. SELECT A NODE WITH MINIMUM AND MAXIMUM DEGREE CONSECUTIVELY (MIN_MAX)

The main function of this algorithm is to find the maximum independent set of any graph, which might be used, among many other services in the graph coloring. If we want to find all Independent Set's, after finding the first MIS, we eliminate the nodes that are in MIS and their edges from the graph, and call the procedure again.

In Fig.4 we apply this algorithm on a graph in Fig.1 and the idea of the algorithm is as the following:

1. Select the node that has minimum degree first and put it in max_ind_set[], as you see there is tow node has the same degree, so we select node that has ascending order [node 2].
2. Delete the selected node and its neighbors from the graph.
3. After deleting the graph containing tow nodes [3,5], apply algorithm on this sub graph to select another node, after applying the algorithm the Maximum Independent Set is [2,3].
4. Select the node that has maximum degree and put it in max_ind_set[], as you see there is three node has the same degree, so we select node that has ascending order [node 1].
5. Delete the selected node and its neighbors from the graph.
6. After deleting the degree of the graph is zero so the second independent set (IS) is node [1].
7. Select this node from the graph, and apply the algorithm by select the node that has minimum degree [node 4]. And delete it with its neighbors. Third (IS) is [4].
8. Select this node from the graph, and apply the algorithm by select the node that has maximum degree. [node 5]. And delete it with its neighbors. Fourth (IS) is [5].

When the algorithm finish, we will obtain on four colors for the entire main graph:

1: [2,3] MIS.
2: [1].
3: [4].
4: [5].

The following algorithm which summarized the above discussion consists of three procedures:

1. Degree (): return zero if there are no nodes in G.
2. select_min_node (): return vertex which has smallest degree.
3. Select_max_node (): return vertex which has largest degree.

Procedure Min_Max:
Input: G(V,E)
Output: MIS
/*Global: int max_ind_set[], char visite_node[], select_min_node, select_max_node;
Get_ind_set() /* function to get the independent sets {
While degree_node( G ) <>0 /* return zero if there are no nodes {
    if g is odd /* variable to determine which procedure select (min or max) {
        v:=select_min_node( G ) /* find the node that has largest degree , if there are more than one choose the first one
    G:=G-v-neighbors(v);
    Visite_node[V]=T
    max_ind_set=max_ind_set U {v}
}Else {
    v:=select_max_node( G ) /* find the node that has largest degree , if there are more than one choose the first one
    G:=G-v-neighbors(v);
    Visite_node[V]=T
    max_ind_set=max_ind_set U {v}
} } */Global: int max_ind_set[], char visite_node[], select_min_node, select_max_node;
Get_ind_set() /* function to get the independent sets {
While degree_node( G ) <>0 /* return zero if there are no nodes {

if g is odd /* variable to determine which procedure select (min or max) 
    
    v:=select_min_node( G ) /* find the node that has largest degree , if there are more than one choose the first one 
    G:=G-v-neighbors(v); 
    Visite_node[V]=T 
    max_ind_set=max_ind_set U {v} 
    } 
    Else 

    v:=select_max_node ( G ) /* find the node that has largest degree , if there are more than one choose the first one 
    G:=G-v-neighbors(v); 
    Visite_node[V]=T 
    max_ind_set=max_ind_set U {v} 
    } 

V.COMPUTATIONAL RESULTS

The three algorithms are programmed in Visual C++ 6.0, compiled and executed on an Intel Pentium 4 CPU 1400 MHZ, 128 RAM. under MS Win XP. The computational results show the difference in performance between the algorithms in terms the CPU run time and the size of graph. The test graph is generated randomly by choosing the size (N) of the graph and the density (D). The size of the graph is varied between 20 up to 5000 at different densities (0.1, 0.2,..., 0.9). 10 graphs are generated and considered their average result. 

In table (1 to 3) we have the computational results for testing the random graphs of three algorithms (SNMD, SNXD, Min_Max ), each with different size 20,...5000 and different densities 0.1,05,0.9. The table's shows size of graphs size of the MIS's, CPU run time performance for finding the MIS. The variation of CPU time between the three algorithm is shown in (figure 1.1 to 1.9) at different densities (0.1,..., 0.9) respectively. In fact the CPU time form density 0.1,...0.4 is smaller in Min_Max algorithm, SNXD at node 3000 than SNMD, but when density is 0.5,...0.9 the SNMD is smaller time. 

In figure (2.1, 2.2 and 2.3) shows the variation between the MIS and size of graph and different densities 0.1, ..., 0.9, as you see the number of nodes in MIS in Min_Max algorithm and SNMX is equal, but in SNXD the number of nodes in MIS is smaller. In figure (3.1,..., 3.3 ) shows the variation time for MIS and the graph size with different densities 0.1,...0.9 ,the SNXD and Min_Max algorithm is more efficient form SNMD to find the MIS with less time. 

In figure (4.1,4.5 and 4.9) shows the relation between the graph size and number of colors with different densities 0.1,...0.9. The number of colors in Min_Max algorithm and SNXD is less small than SNMD. In figure (5.1 and 5.1.1) shows the relation between the CPU time and different density in Min_Max algorithm. Figure (5.2 and 5.2.1) shows the relation between the CPU time and different density in SNMD and figure (5.3 and 5.3.1) shows the relation between the CPU time and different density in SNXD. In fact the CPU time form density 0.1,...0.4 is smaller in Min_Max algorithm, SNXD at node 3000 than SNMD, but when density is 0.5,...0.9 the SNMD is smaller time. 

In figure (6.1,6.2,6.3,6.4) shows there relation between density and the time for MIS, number of color, CPU time MIS in Min_Max algorithm at graph size 5000 with different densities 0.1,...,0.9 . 

In figure (7.1, 7.2, 7.3, 7.4) shows there relation between density and the time for MIS, number of color, CPU time MIS SNX algorithm at graph size 5000 with different densities 0.1,...,0.9 . As mentioned the time is decrease when the density is increase, the MIS is decrease and the number of coloring is increase, but the CPU time is increase and decrease depends on the density. This result is continuously rising by increasing the size of the graph. 

As mentioned the time is decrease when the density is increase, the MIS is decrease and the number of coloring is increase, but the CPU time is increase and decrease depends on the density, as also in figure (8.1, 8.2, 8.3, and 8.4) for SNXD. 

MIS and the minimum number of colors required to color a graph is at small density in general. The time needed to find the MIS is very small relatively at 0.8 or 0.9 densities. 

For further researches plans to compare the Min_Max algorithm with other exact algorithms are suggested. Plans are also recommended to verify that the selection node is always better when it has the highest degree.

VI.CONCLUSION

In this paper we introduce an approximate algorithm to color graph based on MIS finding. We have compared the Min_Max algorithm, SNMD, SNXD. Experimentation proved that that the Min_Max algorithm and SNXD is more efficient than the SNMD algorithm in terms of the number of nodes in a graph and the time complexity. The worst case for the MIS and the minimum number of colors required to color a graph is at small density in general. The time needed to find the MIS is very small relatively at 0.8 or 0.9 densities. 

For further researches plans to compare the Min_Max algorithm with other exact algorithms are suggested.
Plans are also recommended to verify that the selection node is always better when it has the highest degree.

REFERENCES


# Appendix

## TABLE I Average Time in second Min_Max Algorithm, SNMD and SNXD Graph size = (20, 40,.....5000), Density = (0.1)

<table>
<thead>
<tr>
<th>Num of Node</th>
<th>Density</th>
<th>Min_Max Algorithm</th>
<th>SNMD</th>
<th>SNXD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MIS Time(sec)</td>
<td>MIS Time(sec)</td>
<td>MIS Time(sec)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MIS All Color</td>
<td>MIS All Color</td>
<td>MIS All Color</td>
</tr>
<tr>
<td>20</td>
<td>0.1</td>
<td>11 0 0 0 3</td>
<td>11 0 0 0 3</td>
<td>10 0 0 3</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>19 0 0 0 5</td>
<td>19 0 0 0 5</td>
<td>12 0 0 5</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>24 0 0 0 6</td>
<td>24 0 0 0 6</td>
<td>15 0 0 6</td>
</tr>
<tr>
<td>80</td>
<td></td>
<td>27 0 0 0 7</td>
<td>27 0 0 0 7</td>
<td>17 0 0 7</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td>32 0 0 0 8</td>
<td>32 0 0 0 8</td>
<td>18 0 0 8</td>
</tr>
<tr>
<td>120</td>
<td></td>
<td>35 0 0 0 10</td>
<td>35 0 0 0 10</td>
<td>20 0 0 10</td>
</tr>
<tr>
<td>200</td>
<td></td>
<td>45 0 0 0 14</td>
<td>45 0 0 0 14</td>
<td>33 0 0 13</td>
</tr>
<tr>
<td>400</td>
<td></td>
<td>56 0 0 0 22</td>
<td>56 0 0 0 22</td>
<td>36 0 0 22</td>
</tr>
<tr>
<td>600</td>
<td></td>
<td>68 0 0 0 38</td>
<td>68 0 0 0 38</td>
<td>40 0 0 36</td>
</tr>
<tr>
<td>800</td>
<td>0.1</td>
<td>72 0.45 0 0 44</td>
<td>72 0.46 0 0 48</td>
<td>39 0 0 36</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>87 1.763 0 0 81</td>
<td>87 1.772 0 0 87</td>
<td>48 0 0 43</td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td>96 4.906 0 0 110</td>
<td>96 4.176 0 0 124</td>
<td>50 0 0 43</td>
</tr>
<tr>
<td>4000</td>
<td></td>
<td>96 6.691 0 0 139</td>
<td>96 6.75 0 0 153</td>
<td>53 0 0 126</td>
</tr>
<tr>
<td>5000</td>
<td></td>
<td>100 10.935 0 0 165</td>
<td>100 29.731 0 0 296</td>
<td>146 0 0 147</td>
</tr>
</tbody>
</table>

## TABLE II Average Time in second Min_Max Algorithm, SNMD and SNXD Graph size = (20, 40,.....5000), Density = (0.5)

<table>
<thead>
<tr>
<th>Num of Node</th>
<th>Density</th>
<th>Min_Max Algorithm</th>
<th>SNMD</th>
<th>SNXD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MIS Time(sec)</td>
<td>MIS Time(sec)</td>
<td>MIS Time(sec)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MIS All Color</td>
<td>MIS All Color</td>
<td>MIS All Color</td>
</tr>
<tr>
<td>20</td>
<td>0.5</td>
<td>5 0 0 0 8</td>
<td>5 0 0 0 8</td>
<td>8 0 0 8</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>9 0 0 0 13</td>
<td>9 0 0 0 13</td>
<td>13 0 0 10</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>9 0 0 0 16</td>
<td>9 0 0 0 18</td>
<td>18 0 0 13</td>
</tr>
<tr>
<td>80</td>
<td></td>
<td>10 0 0 0 22</td>
<td>10 0 0 0 24</td>
<td>24 0 0 24</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td>10 0 0 0 26</td>
<td>10 0 0 0 29</td>
<td>29 0 0 29</td>
</tr>
<tr>
<td>120</td>
<td></td>
<td>10 0 0 0 30</td>
<td>10 0 0 0 31</td>
<td>31 0 0 31</td>
</tr>
<tr>
<td>200</td>
<td></td>
<td>11 0 0 0 43</td>
<td>11 0 0 0 43</td>
<td>43 0 0 43</td>
</tr>
<tr>
<td>400</td>
<td></td>
<td>13 0.01 0 0 79</td>
<td>13 0.01 0 0 82</td>
<td>7 0 0 21 79</td>
</tr>
<tr>
<td>800</td>
<td></td>
<td>15 0.05 0 159 145</td>
<td>15 0.05 0 159 154</td>
<td>154 0 0 140</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>16 0.08 0 304 179</td>
<td>16 0.08 0 325 196</td>
<td>196 0 0 165</td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td>16 0.15 0 27.7 322</td>
<td>16 0.15 0 24.8 347</td>
<td>347 0 0 302</td>
</tr>
<tr>
<td>4000</td>
<td></td>
<td>18 0.691 0 65.26 476</td>
<td>18 0.701 0 64.423 509</td>
<td>509 0 0 435</td>
</tr>
<tr>
<td>5000</td>
<td></td>
<td>18 1.182 0 142.08 593</td>
<td>18 1.202 0 142.856 628</td>
<td>628 0 0 555</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19 4.912 0 282.11 713</td>
<td>19 2.844 0 273.663 742</td>
<td>742 0 0 657</td>
</tr>
</tbody>
</table>

## TABLE III Average Time in second Min_Max Algorithm, SNMD and SNXD Graph size = (20, 40,.....5000), Density = (0.9)

<table>
<thead>
<tr>
<th>Num of Node</th>
<th>Density</th>
<th>Min_Max Algorithm</th>
<th>SNMD</th>
<th>SNXD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MIS Time(se)</td>
<td>MIS Time(se)</td>
<td>MIS Time(se)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MIS All Color</td>
<td>MIS All Color</td>
<td>MIS All Color</td>
</tr>
<tr>
<td>20</td>
<td>0.9</td>
<td>3 0 0 0 15</td>
<td>3 0 0 0 15</td>
<td>15 0 0 4</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>3 0 0 0 25</td>
<td>3 0 0 0 25</td>
<td>25 0 0 22</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>3 0 0 0 36</td>
<td>3 0 0 0 36</td>
<td>36 0 0 31</td>
</tr>
<tr>
<td>80</td>
<td></td>
<td>4 0 0 0 48</td>
<td>4 0 0 0 47</td>
<td>47 0 0 10</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td>4 0 0 0 55</td>
<td>4 0 0 0 57</td>
<td>57 0 0 19</td>
</tr>
<tr>
<td>120</td>
<td></td>
<td>4 0 0 0 67</td>
<td>4 0 0 0 69</td>
<td>69 0 0 29</td>
</tr>
<tr>
<td>200</td>
<td></td>
<td>4 0 0 0 101</td>
<td>4 0 0 0 102</td>
<td>102 0 0 100</td>
</tr>
<tr>
<td>400</td>
<td></td>
<td>5 0 0.22 185</td>
<td>5 0 0.16 185</td>
<td>185 0 0 98</td>
</tr>
<tr>
<td>800</td>
<td></td>
<td>5 0.01 1.453 339</td>
<td>5 0.01 1.092 362</td>
<td>362 0 0 341</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>5 0.02 2.734 417</td>
<td>5 0.023 2.033 447</td>
<td>447 0 0 220</td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td>6 0.06 19.327 782</td>
<td>6 0.065 13.9 806</td>
<td>806 0 0 780</td>
</tr>
<tr>
<td>3000</td>
<td></td>
<td>6 0.14 61.909 1148</td>
<td>6 0.19 44.544 1187</td>
<td>1187 0 0 1138</td>
</tr>
<tr>
<td>4000</td>
<td></td>
<td>6 0.241 142.17 1464</td>
<td>6 0.254 100.354 1494</td>
<td>1494 0 0 1462</td>
</tr>
<tr>
<td>5000</td>
<td></td>
<td>7 0.3 266.88 1760</td>
<td>7 0.313 189.333 1782</td>
<td>1782 0 0 1777</td>
</tr>
</tbody>
</table>