Parallel Graph Colouring Based on Saturated Degree Ordering

Ahmad Sharieh and Khair Eddin Sabri *

Received on Nov. 5, 2006 Accepted for publication on June 10, 2007

Abstract

The use of graph colouring is essential to many applications such as the estimation of sparse Jacobins, scheduling and registering allocation. There are many existing sequential and parallel algorithms used in graph colouring. A parallel graph colouring algorithm usually requires communication between processors or re-colouring conflict nodes. As the communication between processors increases the time efficiency decreases. Also, when the number of conflict nodes increases the time efficiency decreases and the total number of colours increases. In this paper, we introduce a new parallel graph colouring based on saturated degree ordering (SDO). This algorithm and the one introduced by Gebremedhin and Manne (GM), both based on SDO, were implemented on a single processor and simulated a parallel processing using multiple threads. The algorithms were tested on graphs of different sizes, densities and number of processors. The time required for colouring and the number of colours were measured. Both algorithms decrease their running time as the number of threads increases. In case of sparse graphs, the results indicate that the introduced algorithm is more efficient than the GM algorithm. Also, it was found that the proposed algorithm used less number of colours for the tested samples.

Keywords: Graph algorithms, Heuristic parallel graph colouring, Threads.

Introduction

Colouring of a graph is one of the most useful models in graph theory. This model has been used to solve problems such as school timetabling, computer register allocation, electronic bandwidth allocation, numerical computation, and optimization [5].

Assume you have a graph \( G = (V, E) \), where \( V \) is the set of all nodes and \( E \) is the set of all edges in \( G \). The colouring of a graph \( G \) is a mapping \( C(V): G \to S \), where \( S \) is a finite set of “colours”, such that if \( (v, w) \) is an edge in \( G \), that is if \( (v, w) \in E \), then \( C(v) \neq C(w) \). In other words, adjacent vertices are not assigned the same colour [4]. This leads to the question: can we colour a graph where no adjacent vertices have the same colour by using the fewest number of colours?

Sometimes it is necessary to perform the colouring of a graph on multiple processors system. This will speed up the process of colouring and manipulate graphs.

© 2008 by Yarmouk University, Irbid, Jordan.
* Department of Computer Science, University of Jordan, Amman, Jordan.
Sharieh and Sabri

with large number of vertices. The system can be multi-computers (loosely coupled
distributed processing elements) or multiprocessors (tightly coupled with shared
memory)[8]. Colouring graphs in parallel introduces problems regarding
communication, synchronization, or re-colouring conflict nodes. In this paper, we
introduce a new algorithm to colour a graph in parallel, where the processors can run
concurrency without having conflict nodes to be coloured.

In Section 2, several graph colouring algorithms are reviewed. In Section 3, the
proposed algorithm is introduced. In Section 4, experiments and results are presented.
Section 5 concludes this work.

Review

Graph coloring algorithms are implemented to color a graph sequentially or
concurrency. There are many heuristic sequential techniques for coloring a graph. One of
them is greedy graph colouring [5]. The greedy colouring is heuristic and concentrates
on carefully picking the next vertex to be colored. In the heuristic technique, once a
vertex is colored its color never changes.

The First Fit (FF) algorithm assigns, sequentially, each vertex the lowest legal
colour. It is the easiest and fastest of all heuristic greedy colouring algorithms. This
algorithm has the advantages of being very simple and fast [2].

The Degree-Based Ordering (DBO) algorithm uses a certain selection criterion for
choosing the vertex to be coloured. It picks up a vertex from an arbitrary order. It is a
better strategy than the FF to colour a graph. Many strategies for selecting the next
vertex to be colored have been proposed. Some of the strategies are: Largest Degree
Ordering (LDO), Saturation Degree Ordering (SDO), and Incidence Degree Ordering
(IDO).

The LDO chooses a vertex with the highest number of neighbors. Intuitively, the
LDO provides a better colouring than the FF technique. This heuristic technique can be
implemented to run in O(|E|), which is bounded by O(n^2), in terms of worst case
complexity running time, where n is the number of nodes in the graph [2].

The saturation degree of a vertex is defined as the number of its adjacent vertices of
different colors. The SDO is heuristic approach provides a better colouring than LDO.
This approach can be implemented to run in O(|E|) [2].

A modification of the SDO heuristic is the incidence degree ordering algorithm.
The incidence degree of a vertex is defined as the number of its adjacent coloured
vertices. This heuristic can be implemented to run in O(n^2), in worst case nodes [2].

There are two types of parallel graph colouring based on graph partitioning
approach and block partitioning approach [7]. The graph partitioning approach assumes
that the vertices of a graph G = (V, E) are partitioned into p sets as {V1, ..., Vp}. The set
of shared edges E^S is defined as the set of edges E^S ⊆ E, where (v, w) ∈ E^S and P(v) ≠
P(w). Let the set of shared vertices be the set V^S, where a vertex is in this set if and only
if it is an endpoint for some edge in E^S. Let the set of local vertices in a processing
Parallel Graph Colouring Based on Saturated Degree Ordering

element, denoted by \( V^k \), be the set \( V - V^k \). The \( V^k \) vertices can be coloured, sequentially, using one of the local graph colouring.

The problem is in the \( V^S \) vertices, where there are adjacent vertices located on different processors. So, we need here synchronization between them. There are many heuristic parallel graph coloring algorithms to resolve this problem such as: Jones-Plassmann (JP), Largest Degree First (LDF), and Smallest Degree First (SDF) algorithms. The JP is an asynchronous heuristic parallel and is proposed for distributed memory parallel systems. The proposed heuristic is based on a message-passing model [3]. Jones & Plassman first partition the vertices of the input graph across \( p \) processors in a distributed memory parallel computer. Then, they apply a coloring approach which consists of two phases [3]:

1) An initial parallel phase, which resolves the problem of adjacent vertices on different processors

2) Local phase that uses some good heuristic sequential colouring technique to colour the local vertices of each processor.

At the beginning of the algorithm, each vertex is assigned a random number, and then coloring is performed in parallel depending on the random numbers.

The LDF algorithm is basically similar to the JP algorithm. The only difference is that instead of using random weights to create the independent sets, each weight is chosen to be the degree of the vertex in the graph induced by the uncolored vertices. Random numbers are only used to resolve conflicts between neighbouring vertices having the same degree. Vertices are thus not colored in random order, but rather in order of decreasing degree. The LDF aims to use fewer colors than the JP algorithm [1].

The SDO algorithm tries to improve the LDF algorithm by using a more sophisticated system of weights. In order to achieve this, the algorithm operates in two phases: a weighting phase and a coloring phase. The weighting phase begins by finding all vertices with degree equal to the smallest degree \( d \), presently in the graph. These vertices are assigned the current weight and removed from the graph, thus changing the degree of their neighbours. When there are no vertices of degree \( d \) left, the algorithm looks for vertices of degree \( d+1 \). This process continues until all vertices are assigned weights [1].

The Block Partitioning Approach strategy is to solve the original problem in two phases. In the first phase, the input vertex set is divided into \( p \) blocks of equal sizes. The vertices in each block are colored in parallel using \( p \) processors, where they operate in a synchronous manner. In doing so, two processors may simultaneously attempt to color adjacent vertices leading to an invalid coloring. In the second phase, the vertex set is block partitioned as in the first phase and each processor checks whether or not its vertices are legally colored. If a conflict is discovered, one of the endpoints of the edge, in conflict, is stored in a table. Finally the vertices in this table are colored sequentially [2].

Another strategy of three phases was introduced in [6]. In the first phase, the input vertex set \( V \) of the graph \( G \) is partitioned into \( p \) blocks (of \( V/p \) vertices each). The
vertices in each block are then colored in parallel using \( p \) processors. The parallel coloring comprises of \( k = p \) parallel steps with synchronization barriers at the end of each step. When coloring a vertex, all its previously colored neighbors, both local ones and those found on other blocks, are taken into account. In doing so, two processors may simultaneously attempt to color vertices that are adjacent to each other. If these vertices are given the same color, the resulting coloring becomes invalid and called a pseudo coloring. In the second phase, each processor \( p_i \) checks whether vertices in \( V_i \) are assigned valid colors by comparing the color of each vertex against all its neighbors that were colored at the same parallel step in the first phase. This checking step is also done in parallel. If a conflict is discovered, one of the endpoints of the edge in conflict is stored in a table. Finally, in the third phase, the vertices stored in this table are colored sequentially.

The Proposed Algorithm

The disadvantage of the graph partitioning algorithms is that it takes more time in colouring the \( V^2 \) vertices (first phase), as some processors become idle for a long time, waiting for other processors to color their vertices. Another disadvantage of the block partitioning algorithms is that it recolors conflict vertices sequentially, so it will take more time to recolor a large number of such vertices. However, in the algorithm proposed by Gebremedhin and Manne (GM) [6], the number of conflict nodes is minimized. Therefore, the time used to recolor the nodes is minimized.

Our proposed algorithm uses the SDO algorithm to colour the nodes. It has a technique based on block partitioning that aims at deleting the conflict nodes, so there is no need to recolor them. Therefore, the total number of colors and the time used in coloring should be minimized.

The proposed algorithm works as shown in Figure 1. The total number of processors are \( p \). Each \( P_i, i = 1, 2, \ldots, (p-1) \), chooses the next node to be colored and \( P_c \) colors the selected nodes for coloring. The main steps in the proposed algorithm are shown in Figure 2. The example in Figure 3 shows a snapshots of colouring a graph.
Partition the graph $G=\langle V,E \rangle$ into $(p-1)$ blocks and assign each block to a processor.

For each $P_i$ (Search-Processor), do in parallel:

1. Call SDO algorithm to select the next node to be coloured.
2. Flag the selected nodes for $P_c$ to be coloured.
3. $P_c$ (Colouring-Processor) applies the colouring algorithm to colour the flagged nodes. This algorithm colours the selected nodes with the minimum possible colour which is different from the color of all adjacent nodes to this received node.
4. Repeat the process in 2.1 to 2.3 until all nodes have been coloured.

End.

**Figure 1:** Diagram of the proposed algorithm

**Figure 2:** The major steps in the proposed colouring algorithm corresponding to Figure 1.

To colour a graph in parallel, using $p$ processors, $(p-1)$ processors are used to search for the next nodes to be coloured and one processor is used in colouring. The graph is partitioned into $(p-1)$ blocks. The sizes of the blocks are almost equal. Each block is assigned to a processor to colour the nodes in this block. The function of each processor holding a block is to search for the next node to be coloured. The colouring process for the selected nodes is performed by another processor $P_c$, specialized for colouring. Therefore, the colouring step is done in parallel with other processors that search for the best nodes to be coloured. The $P_c$ has information kept in its site about the previously coloured nodes. It can use any sequential colouring algorithm such as: LDO, SDO, and IDO. In this research, the $P_c$ employs the SDO algorithm. It needs only to colour new nodes from other processors, thus conflicts is removed.

The worst case run time of the SDO algorithm is bounded by $O(n^2)$, where $n$ is the total number of nodes. Most of the time is needed in searching for the best nodes to be coloured. The colouring steps take no more than $O(n*c)$, where $c$ is the total number of colours.

The run time cost of the proposed algorithm consists of: time in step 1, time in steps 2.1 and 2.2 for the first round, time in steps 2.1, 2.2 and 2.3 in round two to last round minus 1, and time to colour nodes in the last round by $P_c$. The run time in performing
step 2.1 and 2.2 is the cost of searching for the candidate to be coloured in each block, using the SDO. This time is bounded by $O(n^2/(p-1) + (p-1))$ of computation (searching and assigning flags) time. The cost in step 2.3 is bounded by $O(k*(p-1))$, where $k$ is the maximum number of colours needed to colour the $p-1$ flagged nodes in a round. The colouring process in step 2.3 is performed concurrently with that in step 2.1 and 2.2. The total run time can be approximated by $O(n^2/(p-1)) + O(p-1) + O(k*(p-1))$. This is in the worst case. The run time complexity is dominated by $O(n^2/(p-1))$. Note that the first round will take the longest time to colour first set of nodes. Then, the number of nodes to select a candidate from and the number of nodes to colour are decreased at the end of each round. This will speed up the process of colouring the graph.
Parallel Graph Colouring Based on Saturated Degree Ordering

Figure 3: Snapshots of colouring a graph.
The complexity of the sequential algorithm (SDO) is bounded by $O(n^2 + c*n)$, where $n$ is the total number of nodes and $c$ the total number of colours. The worst case is when the graph is fully connected, so we have $c=n$. Therefore, the worst case is $O(2^n)$.

The SDO in step 2.1 can be replaced by a more efficient algorithm if there is one.

For a system with shared memory, the blocks will be loaded once and the time for communication is replaced by the time for synchronization. The searching-processors work on their blocks independently. The synchronization is between each of the searching-processors and the colouring-processor $P_c$. Thus, the algorithm will perform better in case of using multi-processors system. This is what we investigated in this work.

**Result and Analysis**

The proposed algorithm has been implemented using Java programming language and tested on a single processor computer (PC). Threads were used to represent processors. A similar implementation and testing of the previous algorithm due to Gebremedhin and Manne [6] was provided. The experimental results from the GM and the proposed algorithm are reported. Both implementations were based on the SDO. Different graphs, generated randomly, have been used.

The two algorithms were tested on different graphs, using more than one processor and different densities. Samples of the results are shown in Table 1. Three different sets of graphs with 500, 1000, 1500 nodes are presented here. Three different densities 10%, 50%, and 90% are used. The algorithms were tested by using different number of threads: 1 to 9 threads. The same experiments were repeated ten times and the average were taken. This is to smoothen out the effects of the random numbers. The following results were obtained:

**Table 1:** Time and Number of colors for coloring different graphs with different densities by the proposed algorithm and the GM algorithm.

<table>
<thead>
<tr>
<th>Line</th>
<th>no of processors</th>
<th>Connection Density %</th>
<th>no of nodes</th>
<th>Proposed Algorithm Time</th>
<th>No. Of colors</th>
<th>Gebremedhin &amp; Manne Alg. Time</th>
<th>No. Of colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
<td>500</td>
<td>139</td>
<td>16</td>
<td>139</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>10</td>
<td>500</td>
<td>141</td>
<td>16</td>
<td>50</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>10</td>
<td>500</td>
<td>43</td>
<td>18</td>
<td>34</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>10</td>
<td>500</td>
<td>28</td>
<td>19</td>
<td>28</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>10</td>
<td>500</td>
<td>25</td>
<td>19</td>
<td>26</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>10</td>
<td>500</td>
<td>19</td>
<td>19</td>
<td>28</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>10</td>
<td>500</td>
<td>20</td>
<td>19</td>
<td>27</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>10</td>
<td>500</td>
<td>26</td>
<td>19</td>
<td>28</td>
<td>19</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>10</td>
<td>500</td>
<td>23</td>
<td>19</td>
<td>30</td>
<td>19</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>50</td>
<td>500</td>
<td>1128</td>
<td>68</td>
<td>1128</td>
<td>68</td>
</tr>
</tbody>
</table>

496
Parallel Graph Colouring Based on Saturated Degree Ordering

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>2</td>
<td>50</td>
<td>500</td>
<td>1128</td>
<td>68</td>
<td>331</td>
<td>70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>50</td>
<td>500</td>
<td>326</td>
<td>70</td>
<td>159</td>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>4</td>
<td>50</td>
<td>500</td>
<td>165</td>
<td>71</td>
<td>123</td>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>5</td>
<td>50</td>
<td>500</td>
<td>114</td>
<td>72</td>
<td>97</td>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>6</td>
<td>50</td>
<td>500</td>
<td>87</td>
<td>71</td>
<td>83</td>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>7</td>
<td>50</td>
<td>500</td>
<td>75</td>
<td>72</td>
<td>78</td>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>8</td>
<td>50</td>
<td>500</td>
<td>69</td>
<td>72</td>
<td>75</td>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>9</td>
<td>50</td>
<td>500</td>
<td>67</td>
<td>72</td>
<td>69</td>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>90</td>
<td>500</td>
<td>1345</td>
<td>167</td>
<td>1345</td>
<td>167</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>90</td>
<td>500</td>
<td>1346</td>
<td>167</td>
<td>428</td>
<td>174</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>3</td>
<td>90</td>
<td>500</td>
<td>439</td>
<td>174</td>
<td>212</td>
<td>176</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td>90</td>
<td>500</td>
<td>203</td>
<td>176</td>
<td>145</td>
<td>177</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>5</td>
<td>90</td>
<td>500</td>
<td>144</td>
<td>176</td>
<td>113</td>
<td>176</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>6</td>
<td>90</td>
<td>500</td>
<td>115</td>
<td>176</td>
<td>94</td>
<td>175</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>7</td>
<td>90</td>
<td>500</td>
<td>100</td>
<td>176</td>
<td>84</td>
<td>177</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>8</td>
<td>90</td>
<td>500</td>
<td>91</td>
<td>175</td>
<td>82</td>
<td>178</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>9</td>
<td>90</td>
<td>500</td>
<td>81</td>
<td>177</td>
<td>73</td>
<td>176</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>1</td>
<td>10</td>
<td>1000</td>
<td>957</td>
<td>30</td>
<td>957</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>2</td>
<td>10</td>
<td>1000</td>
<td>962</td>
<td>30</td>
<td>311</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>3</td>
<td>10</td>
<td>1000</td>
<td>308</td>
<td>30</td>
<td>175</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>4</td>
<td>10</td>
<td>1000</td>
<td>172</td>
<td>30</td>
<td>135</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>5</td>
<td>10</td>
<td>1000</td>
<td>119</td>
<td>31</td>
<td>118</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>6</td>
<td>10</td>
<td>1000</td>
<td>101</td>
<td>31</td>
<td>103</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>7</td>
<td>10</td>
<td>1000</td>
<td>84</td>
<td>31</td>
<td>95</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>8</td>
<td>10</td>
<td>1000</td>
<td>78</td>
<td>31</td>
<td>92</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>9</td>
<td>10</td>
<td>1000</td>
<td>75</td>
<td>31</td>
<td>94</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>1</td>
<td>50</td>
<td>1000</td>
<td>8351</td>
<td>116</td>
<td>8351</td>
<td>116</td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>2</td>
<td>50</td>
<td>1000</td>
<td>8584</td>
<td>116</td>
<td>2392</td>
<td>124</td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>3</td>
<td>50</td>
<td>1000</td>
<td>2431</td>
<td>125</td>
<td>1261</td>
<td>126</td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>4</td>
<td>50</td>
<td>1000</td>
<td>1258</td>
<td>125</td>
<td>819</td>
<td>126</td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>5</td>
<td>50</td>
<td>1000</td>
<td>823</td>
<td>125</td>
<td>637</td>
<td>126</td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>6</td>
<td>50</td>
<td>1000</td>
<td>629</td>
<td>126</td>
<td>514</td>
<td>127</td>
<td></td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>7</td>
<td>50</td>
<td>1000</td>
<td>505</td>
<td>126</td>
<td>459</td>
<td>127</td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>8</td>
<td>50</td>
<td>1000</td>
<td>433</td>
<td>125</td>
<td>430</td>
<td>126</td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>9</td>
<td>50</td>
<td>1000</td>
<td>377</td>
<td>126</td>
<td>384</td>
<td>126</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>1</td>
<td>90</td>
<td>1000</td>
<td>10383</td>
<td>304</td>
<td>10383</td>
<td>304</td>
<td></td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>2</td>
<td>90</td>
<td>1000</td>
<td>10187</td>
<td>304</td>
<td>3136</td>
<td>317</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

497
1000 3126 319 1723 320
1679 317 1158 321
1164 318 830 319
878 319 659 321
732 319 570 320
645 319 495 320
590 319 450 321

206 41 241 42

27917 163 27917 163
28922 164 7810 174
7958 174 4086 176
4012 175 2678 176
2649 175 2039 176
1964 175 1570 177
1386 175 1183 176
1191 175 1119 176

34606 433 34606 433
34660 433 10230 452
10250 452 5358 452
5328 452 3592 455
3598 452 2733 457
2939 452 2130 456
2428 454 1830 455
2103 452 1522 455
1881 452 1353 454

<table>
<thead>
<tr>
<th>53</th>
<th>3</th>
<th>90</th>
<th>1000</th>
<th>3126</th>
<th>319</th>
<th>1723</th>
<th>320</th>
</tr>
</thead>
<tbody>
<tr>
<td>54</td>
<td>4</td>
<td>90</td>
<td>1000</td>
<td>1679</td>
<td>317</td>
<td>1158</td>
<td>321</td>
</tr>
<tr>
<td>55</td>
<td>5</td>
<td>90</td>
<td>1000</td>
<td>1164</td>
<td>318</td>
<td>830</td>
<td>319</td>
</tr>
<tr>
<td>56</td>
<td>6</td>
<td>90</td>
<td>1000</td>
<td>878</td>
<td>319</td>
<td>659</td>
<td>321</td>
</tr>
<tr>
<td>57</td>
<td>7</td>
<td>90</td>
<td>1000</td>
<td>732</td>
<td>319</td>
<td>570</td>
<td>320</td>
</tr>
<tr>
<td>58</td>
<td>8</td>
<td>90</td>
<td>1000</td>
<td>645</td>
<td>319</td>
<td>495</td>
<td>320</td>
</tr>
<tr>
<td>59</td>
<td>9</td>
<td>90</td>
<td>1000</td>
<td>590</td>
<td>319</td>
<td>450</td>
<td>321</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>61</th>
<th>1</th>
<th>10</th>
<th>1500</th>
<th>2991</th>
<th>40</th>
<th>2991</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>62</td>
<td>2</td>
<td>10</td>
<td>1500</td>
<td>3086</td>
<td>40</td>
<td>931</td>
<td>41</td>
</tr>
<tr>
<td>63</td>
<td>3</td>
<td>10</td>
<td>1500</td>
<td>937</td>
<td>41</td>
<td>530</td>
<td>42</td>
</tr>
<tr>
<td>64</td>
<td>4</td>
<td>10</td>
<td>1500</td>
<td>550</td>
<td>41</td>
<td>384</td>
<td>42</td>
</tr>
<tr>
<td>65</td>
<td>5</td>
<td>10</td>
<td>1500</td>
<td>348</td>
<td>41</td>
<td>323</td>
<td>42</td>
</tr>
<tr>
<td>66</td>
<td>6</td>
<td>10</td>
<td>1500</td>
<td>273</td>
<td>41</td>
<td>278</td>
<td>42</td>
</tr>
<tr>
<td>67</td>
<td>7</td>
<td>10</td>
<td>1500</td>
<td>234</td>
<td>41</td>
<td>262</td>
<td>42</td>
</tr>
<tr>
<td>68</td>
<td>8</td>
<td>10</td>
<td>1500</td>
<td>206</td>
<td>41</td>
<td>241</td>
<td>42</td>
</tr>
<tr>
<td>69</td>
<td>9</td>
<td>10</td>
<td>1500</td>
<td>197</td>
<td>41</td>
<td>237</td>
<td>42</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>71</th>
<th>1</th>
<th>50</th>
<th>1500</th>
<th>27917</th>
<th>163</th>
<th>27917</th>
<th>163</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>2</td>
<td>50</td>
<td>1500</td>
<td>28922</td>
<td>164</td>
<td>7810</td>
<td>174</td>
</tr>
<tr>
<td>73</td>
<td>3</td>
<td>50</td>
<td>1500</td>
<td>7958</td>
<td>174</td>
<td>4086</td>
<td>176</td>
</tr>
<tr>
<td>74</td>
<td>4</td>
<td>50</td>
<td>1500</td>
<td>4012</td>
<td>175</td>
<td>2678</td>
<td>176</td>
</tr>
<tr>
<td>75</td>
<td>5</td>
<td>50</td>
<td>1500</td>
<td>2649</td>
<td>175</td>
<td>2039</td>
<td>176</td>
</tr>
<tr>
<td>76</td>
<td>6</td>
<td>50</td>
<td>1500</td>
<td>1964</td>
<td>175</td>
<td>1570</td>
<td>177</td>
</tr>
<tr>
<td>77</td>
<td>7</td>
<td>50</td>
<td>1500</td>
<td>1587</td>
<td>175</td>
<td>1339</td>
<td>176</td>
</tr>
<tr>
<td>78</td>
<td>8</td>
<td>50</td>
<td>1500</td>
<td>1386</td>
<td>175</td>
<td>1183</td>
<td>176</td>
</tr>
<tr>
<td>79</td>
<td>9</td>
<td>50</td>
<td>1500</td>
<td>1191</td>
<td>175</td>
<td>1119</td>
<td>176</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>81</th>
<th>1</th>
<th>90</th>
<th>1500</th>
<th>34606</th>
<th>433</th>
<th>34606</th>
<th>433</th>
</tr>
</thead>
<tbody>
<tr>
<td>82</td>
<td>2</td>
<td>90</td>
<td>1500</td>
<td>34660</td>
<td>433</td>
<td>10230</td>
<td>452</td>
</tr>
<tr>
<td>83</td>
<td>3</td>
<td>90</td>
<td>1500</td>
<td>10250</td>
<td>452</td>
<td>5358</td>
<td>452</td>
</tr>
<tr>
<td>84</td>
<td>4</td>
<td>90</td>
<td>1500</td>
<td>5328</td>
<td>452</td>
<td>3592</td>
<td>455</td>
</tr>
<tr>
<td>85</td>
<td>5</td>
<td>90</td>
<td>1500</td>
<td>3598</td>
<td>452</td>
<td>2733</td>
<td>457</td>
</tr>
<tr>
<td>86</td>
<td>6</td>
<td>90</td>
<td>1500</td>
<td>2939</td>
<td>452</td>
<td>2130</td>
<td>456</td>
</tr>
<tr>
<td>87</td>
<td>7</td>
<td>90</td>
<td>1500</td>
<td>2428</td>
<td>454</td>
<td>1830</td>
<td>455</td>
</tr>
<tr>
<td>88</td>
<td>8</td>
<td>90</td>
<td>1500</td>
<td>2103</td>
<td>452</td>
<td>1522</td>
<td>455</td>
</tr>
<tr>
<td>89</td>
<td>9</td>
<td>90</td>
<td>1500</td>
<td>1881</td>
<td>452</td>
<td>1353</td>
<td>454</td>
</tr>
</tbody>
</table>

71
72
73
74
75
76
77
78
79
81
82
83
84
85
86
87
88
89
Parallel Graph Colouring Based on Saturated Degree Ordering

1. When a small number of processors was used, the GM algorithm worked better than the proposed algorithm. This is because of unbalanced distribution of work that could happen in this case. The $P_x$ has more load than the other processors. This can be shown clearly in line 2, 12, 22, etc. where the number of processors is two.

2. It was found that the proposed algorithm performs better than the GM algorithm if the graph is sparse and the number of processors is more than four, as shown, for example, in lines 4-9, 34-39, 64-69 in Table 1. This is due to the fact that in the proposed algorithm there is no conflict nodes need to be resolved, which usually takes time.

3. If the density of the graph is close to one, the GM algorithm performs better in some cases as shown, for example, in lines 21-29, 51-59, 89-89 in Table 1. This is because in a graph with high density, the load on colouring processor increases.

4. The proposed algorithm produced fewer number of colours than GM did, as shown, for example, in lines 2-9, 32-39, 64-69 in Table 1. This is because in the GM algorithm, the number of conflict nodes increases the number of colours.

We used timing of simulation to justify the proposed algorithm compared to the GM algorithm. The simulation results include the time spent for synchronization communication in both algorithms. The results verify the good quality of the proposed algorithm against the GM algorithm. See Figure 4. The proposed algorithm ended up with smaller chromatic number than the GM algorithm.
Figure 4: comparison between the proposed algorithm and the GM algorithm.
Conclusion

In this paper, we propose a new parallel algorithm for graph coloring. It aims at eliminating the conflict nodes. The main differences from the GM algorithm are: SDO was used instead of FF and one processor is dedicated to do the colouring to avoid conflict. It should be obvious that different degree-based ordering can be used with the block partitioning approach, so the first difference is not very novel. The second difference is the major contribution in this work.

The results have shown that

1. The proposed algorithm is faster than the GM when using more than four processors.
2. The number of colours produced by the proposed algorithm is less than of that produced by GM algorithm in most cases.
3. The proposed algorithm and the GM algorithms show good results in implementing them on one machine with multithreads.

This work can be extended by

1. Implementing and testing the proposed algorithm on real multi-processors machines.
2. Focusing on modifying the proposed algorithm to decrease the time needed to color the nodes, especially if the density of the graph is high. It is suggested that more than one processor is used to color nodes and make synchronization between them.
3. In the proposed algorithm, all the nodes are colored using Pc processor. Alternative work can be done by letting the Pc processor colour the shared nodes only and each processor colour its local nodes.
توظيف مخطط بناءً على الدرجة المشبعة للعقد فيه

أحمد الشريعة و خير الدين صبري

ملخص

عملية تلوين المخططات ضرورية ل كثير من التطبيقات مثل الجدولة والحجز والتعامل مع مصفوفة جيكوب. والعملية تحتاج لوقت طويل لتنفيذها. ويوجد عدد من الخوارزميات المتسلسلة والمتوازية المستخدمة للتلوين، إلا أن الخوارزمية المتوازية تتطلب الاتصال بين المعالجات أو التزام تلوين عدد مشتركة. وكلما زاد زمن التواصل فلت المالية، ويزداد عدد الألوان كلما زادت التعارضات. وفي هذه الورقة تم تقديم خوارزمية متوازية جديدة للتلوين بنية على أساس الدرجة المشبعة للعقد. وتم تنفيذ الخوارزمية المقترحة ومقارنتها مع الخوارزمية المقدمة من جبريسده ومين. وعلى شكل متسلسل ومتفاوض

وباستخدام خطوط المهام المتعددة.

ومثل اختبار الخوارزمية المقترحة للتلوين مخططات بعض من العقد المختلفة وذات كفاءة مختلفة وعلى عدد مختلف من المعالجات. وتم قياس الزمن اللازم لتلوين المخططات وعدد الألوان المستخدمة. وجد أن الخوارزمية المقترحة أفضل في حال كان المخطط غير كثيف من حيث المدة اللازم لتلوين عدد الألوان وجميع العينات التي تم اختبارها.

References


Parallel Graph Colouring Based on Saturated Degree Ordering
