A STUDY ON OPTIMIZING GRAPH COLORING PROBLEM USING PARALLEL ARCHITECTURES

M. Nirmala, Ketan Patel, Nirav Kagathara, Nisarg Patel
School Of Information Technology and Engineering, Vit University, Vellore, Tamilnadu.
Email: nirmaladhinesh@gmail.com

Received on 25-10-2016 Accepted on 02-11-2016

Abstract

Graph coloring is used in many real-time applications, and Graph coloring is an important concept in graph theory. Various areas of computer applications for research use graph coloring ideas. The main aim of this paper is to study on optimizing time because graph is coloring NP-hard problem, so time-consuming process. In this paper, we study a parallel architecture for graph coloring algorithm. We implement in Open MP analyzing proposed parallel algorithm using Intel v-tune analyzer. After analyzing parallel graph coloring algorithm, we got 30% to 32% time optimization in 2-core processor. This can be used in many applications like air traffic control, GPS frequency reallocation, register allocation, timetable scheduling, and modeling sensor network. Various papers based on graph coloring have been studied related to parallel architecture, computer science application, and an overview has been presented here.

Keyword: NP-Hard, Open MP, Graph Coloring

1. Introduction

The development of cloud technology & big data [1-5], revolutionized the information technology to a greater extent. The graph theory plays major role in analyzing the big data. In graph theory, graph coloring may be a special case of graph labeling; it's an assignment of labels historically referred to as colors to parts of a graph, subject to bound constraints. In its simplest kind, it's some way of coloring the vertices of a graph specified no 2 adjacent vertices share a similar color; this can be referred. Similarly, a foothold coloring assigns a color to every edge, so no 2 adjacent edges share a similar color, and a face coloring of a planate graph assigns a color to every face or region so no 2 faces that share a boundary have a similar color [6]. A coloring of a graph is an assignment of a color to each vertex of the graph so that no two adjacent vertices. There are two types of graph coloring. First vertex coloring and second edge coloring and
we are mainly focused on vertex coloring, by default graph coloring is vertex coloring. A chart can be colored by assigning a different color to each of its vertices.

However, we use fewer color to color all vertices than a number of vertices in the graph then we can call the Chromatic number of the graph which is minimum or least number of colors needed for graph or vertex coloring of the graph and denoted by $\chi(G)$ (key of Graph).

Exam scheduling and generating student timetable is challenging task that universities and college face several times every year. By using graph coloring algorithm, we can schedule above two such that no conflict among students, faculty, and courses taken by a student that is no 2 or more exams for the same student are scheduled at the same time. For above two scheduling, we use course as a vertex. Finding a chromatic number of a graph is an NP-hard problem [7], hence we cannot expect to solve it efficiently for the large graph, and we use Welch Powell’s algorithm [8] to find a chromatic number of a graph.

**Sample example of generating of time table:**

In this, we have got to represent this situation in terms of the graph for locating a minimum variety of your day slots or periods required to schedule each course, and here we elect course as a vertex and edge as combine in the fact that creates conflicts and color as a time slots. To demonstrate us to take into account [five subjects as Maths, DBMS, DS, CA, ADA.]

**Figure 1: sample graph**

![Sample Graph](image)

**Figure 2: Adjacency matrix**

<table>
<thead>
<tr>
<th></th>
<th>maths</th>
<th>DBMS</th>
<th>DS</th>
<th>CA</th>
<th>ADA</th>
</tr>
</thead>
<tbody>
<tr>
<td>maths</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBMS</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>AdjMatrix</td>
<td>DS</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>CA</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>ADA</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Represent on top of the graph in matrix illustration that is contiguity matrix is that the computer files for Welch Powell’s algorithmic program and produces output as achromatic range. Here we have a tendency to square measure presenting one in every of the chances of graph coloring taken. From fig- two.1 we will conclude that to schedule five subjects, we have a tendency to needed three slot or periods C1, C2, and C3. In terms of graph minimum three colors square measure necessary to paint all five vertices and [CA and database management system will schedule in C2, Maths and DS in C3 and ADA in C1.]

2. Existing Methodology

WELCH POWELL’S ALGORITHM OR GRAPH COLORING ALGORITHM:

[Generally it follows main two steps, they are

1. organize the vertices in the decreasing order of their degrees, if two or additional vertices have the same level then organize the vertices in alphabetical or numerical order.

2. Assign colors to vertices in higher than order specified no 2 adjacent vertices have the same color.]

To run this rule, we'd like three things area unit vital here

2.1. Realize degree of vertices

2.2. Realize most degree of vertices

2.3. Realize the remaining colors

A. Algorithm for realizing degree of vertices:

degreecount(int n)
{
    Declare i,j,count as integer
    For i=0 to n-1
    {
        dcount=0;
        for j=0 to n-1
        {
            if(graph[i][j]==1) dcount++;
        }
    }
}
degree[i] = count;
}

B. Algorithm for realizing maximum
the degree of vertices:

int Finding_max_degree_vertex(int n)
{
    int i, max_degree, max_degree_index;
    max_degree = 0;
    max_degree_index = 0;
    for (i = 0; i < n; i++)
    {
        if (Degree[i] > max_degree && Flag[i] != 1)
        {
            max_degree = Degree[i];
            max_degree_index = i;
        }
    }
    return max_degree_index;
}

C. Algorithm for realizing the remaining colors:

int Finding_available_colour(int n)
{
    int i, j, count = 0, key, serching_flag;
    for (i = 0; i < n; i++)
    {
        if (Colour_of_vertex[i] != 0)
        {
            key = Colour_of_vertex[i];
            ...
serching_flag = 0;
for (j = 0; j<n; j++){
    if (key == Avaible_colour[j]){ 
        serching_flag = 1;
    }
}
if (serching_flag == 0){
    Avaible_colour[count]= Colour_of_vertex[i];
    count++;
    count++;
}
}
return count;

3. Optimizing Graph Colouring

Open MP: In order to get the most performance get processed from a processor with Hyper-Threading Technology, Associate application has to be dead in parallel. Parallel execution needs threads, Associate in multithreading an application isn't trivial. The designers of Open MP wished to produce a simple methodology to string applications while not requiring that the coverskills make, synchronize, and destroy threads or perhaps requiring him or her to work out what number threads to go. In this approach, we are using a barrier, critical section, and separate function. The Syntax for the:

#program omp parallel for;
#program omp barrier;
#program omp critical;
#program omp section function;
4. Complexity and Analysis

4.1 Complexity

This formula is understood to use the most degree (\(G\)) +1 colors wherever \(d(G)\) represents the most important worth of the degree within the graph \(G\).

In Degree count perform graph is traversed \(V \times V\) times \(O(V^2)\) all told cases In perform to search out the degree vertex all case completeness is \(O(V)\) Finding out there colors \(O(V)\)In Color Schedule perform \(O(V^2)\) to urge out there colors than color it Time Complexity: \(O(V^2 + )\) in worst case.

4.3 Intel VTune Amplifier analysis

4000 vertices Sequential code Analysis, 4000 vertices parallel performance on dual core machine, 200 vertices Sequential code output, and 200 vertices parallel output is depicted in the figure 3, 4, 5, 6, 7 and 8.
Figure 6: Lightweight Hotspot analysis of Parallel Code

Figure 7: sequential output

Figure 8: parallel output

5. Conclusion

Hence sequential & parallel implementation of Welch-Powell Algorithm for graph coloring is done. Here we have used Matrix and array to denote properties of each vertex; we intend to experiment with further, amore powerful data structure that will store multiple attributes corresponding to vertex more efficiently.

References


