Graph Coloring Problem

CS594 Combinatorial Optimization

Prof. John Lillis

Laura Varrenti
SS#3587

Marco Domenico Santambrogio
SS#3587
Graph Coloring Problem

- **Input:** graph $G = (V, E)$
  - $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices
  - $E$ the set of edges connecting the vertices

- **Constraint:** no two vertices can be in the same color class if there is an edge between them

- There are two variants of this problem
  - In the optimisation variant, the goal is to find a colouring with a minimal number of colours, or partition of $V$ into a minimum number $\chi(G)$ of color classes $C_1, C_2, \ldots, C_k$
  - whereas in the decision variant, the question is to decide whether for a particular number of colours, a colouring of the given graph exists
Graph Coloring is NP-hard

It is unlikely that efficient algorithms guaranteed to find optimal colorings exists.

Thus, practically, we develop heuristic algorithms that find near-optimal colorings quickly.
1 Kings 3

11 And God said unto him, Because thou hast asked this thing, and hast not asked for thyself long life; neither hast asked riches for thyself, nor hast asked the life of thine enemies; but hast asked for thyself understanding to discern judgment;

12 Behold, I have done according to thy words: lo, I have given thee a wise and an understanding heart; so that there was none like thee before thee, neither after thee shall any arise like unto thee.

13 And I have also given thee that which thou hast not asked, both riches, and honour: so that there shall not be any among the kings like unto thee all thy days.
Problem Definition

- We want to implement the “Salomone” module

**Input:**
- Threads Dependency Graph
- Time constraints

**Output:**
- Minimum number of slices
- Threads set
We start from a behavioural description:

Process test(p,...)
    in port p[SIZE];
    {
        ...
        v = read p;
        while(v>=0)
        {
            {
                <loop-body>
                v = v-1;
            }
        }
    }
We divide the BHD in Threads:

- **T1** ->
  - perform thereading operation
  - read v
  - deteach

- **T2** ->
  - Consists of operation in the body of the loop
  - loop_sync
  - <loop body>
  - v = v - 1
  - deteach
Threads Dependency Graph

S1

Read

Signal1

Wait2

Wait1

Body

Loop cond

Signal2

S1
UIC

Conflict Graph

Threads Dependency Graph

Time Constraints

Graph Generator Module

Conflict Graph
An example

Threads Dependency Graph
Find the Overlapping Tasks
UIC
Conflict Graph

Diagram of a conflict graph with nodes labeled 1/1, 1/2, 2/2, 2/1, 2/3, 2/4, 3/1, 3/2, 1/3.
A possible solution:
Some GCP Algorithms:

- DSATUR
- BSC
- RLF
Degree of Saturation Algorithm (DSATUR)

- Sequential coloring, dynamically chooses the vertex to color next, picking one that is adjacent to the largest number of distinctly colored vertices.
- Degree of saturation of a vertex \( v \), \( \text{deg}_s(v) \), number of different colors already assigned to the vertices adjacent to \( v \).
- Complexity is \( O(|V|^3) \).
Sequential coloring algorithm

Initially vertices are ordered according to non-decreasing degree, the order is dynamically changed.

Assume $v_1, \ldots, v_{i-1}$ have already been colored using $l_i$ different colors.

Assume the set of free colors for $v_i$ is the subset of colors in $U = \{1, 2, \ldots, l_i + 1\}$, which are not present in the neighborhood of $v$. If an upperbound $opt$ for $\chi(G)$ has been established, all colors $\geq opt$ can be removed from $U$.

The vertex to be colored next is the one of maximal $deg_s$. It is colored with the smallest color in $U$. If $U$ is empty, backtrack is executed.
U = \{\text{Red, Yellow, Blu, Green}\}
It colors the vertices one color at a time, in “greedy” fashion.

- Class $C$ is constructed as follows:
  - $V'$ = set of uncolored vertices
  - $U$ = set of uncolored vertices that cannot be legally placed in $C$
1. Choose $v_0 \in V'$ that has the maximum number of edges to vertices $\in V'$
2. $C \leftarrow \{v_0\}$
3. $U \leftarrow$ all $u \in V'$ that are adjacent to $v_0$
4. $V' \leftarrow V' - U$
5. While ($V' \neq \emptyset$)
   {5.1. Choose $v \in V'$ that has maximum number of edges to vertices in $U$
   5.2. $C \leftarrow C \cup \{v\}$
   5.3. $U \leftarrow U \cup$ all $u \in V'$ adjacent to $v$
   5.4. $V' \leftarrow V' -$ all $u \in V'$ adjacent to $v$
   }
In the worst case time complexity of RLF is $O(|V|^3)$.

One factor $|V|$ is due to determination of vertex $v_0$ of maximal degree.

Traversing all the non-neighbors of $v_0$ searching a vertex $v$ with maximal number of common neighbors with $v_0$ may cost another $O(|V|^2)$ elementary operations.
RLF - Simulation

Graph with nodes A, B, C, D, E, F, G and edges labeled with numbers 1, 2, 3, 4, 5, 6, 7.
A possible problem
### The problem

<table>
<thead>
<tr>
<th>Delay</th>
<th>1/1</th>
<th>1/2</th>
<th>2/2</th>
<th>2/3</th>
<th>2/4</th>
<th>1/3</th>
<th>3/1</th>
<th>3/2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta(1/1)$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta(1/2)$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta(1/3)$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta(2/1)$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta(2/2)$</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta(2/3)$</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta(2/4)$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta(3/1)$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta(3/2)$</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tasks: S1, S2

Problem
Our Conflict Graph
A new possible solution
A new possible solution
Our Aims

- Minimize the number of color in the GCP applied on the conflict graph

- Find all the Threads sets

- Considering the decision variant of the GCP we want to see if a specific number of slice, a particular number of coloro, on an given FPGA is enough to have a coloring solution of the given graph
We decide to use the DIMACS format because, as we can see in their document:

- The DIMACS format is a flexible format suitable for many types of graph and network problems. This format was also the format chosen for the First Computational Challenge on network flows and matchings.
- This is a format for graphs that is suitable for those looking at graph coloring and finding cliques in graphs.
- One purpose of the DIMACS Challenge is to ease the effort required to test and compare algorithms and heuristics by providing a common testbed of instances and analysis tools.
**UIC**  
**DIMACS COLORING BENCHMARKS**

- flat1000_50_0.col.b (1000,245000), 50, **CUL**
- flat1000_60_0.col.b (1000,245830), 60, **CUL**
- flat1000_76_0.col.b (1000,246708), 76, **CUL**
- flat300_20_0.col.b (300,21375), 20, **CUL**
- flat300_26_0.col.b (300, 21633), 26, **CUL**
- flat300_28_0.col.b (300, 21695), 28, **CUL**
- fpsol2.i.1.col (496,11654), 65, **REG**
- fpsol2.i.2.col (451,8691), 30, **REG**
- fpsol2.i.3.col (425,8688), 30, **REG**
- inithx.i.1.col (864,18707), 54, **REG**
- inithx.i.2.col (645, 13979), 31, **REG**
- inithx.i.3.col (621,13969), 31, **REG**
- latin_square_10.col (900,307350), ?, **le450_15a.col** (450,8168), 15, **LEI**
UIC
DIMACS COLORING BENCHMARKS

- DSJC1000.1.col.b (1000, 99258), ?, DSJ
- DSJC1000.5.col.b (1000, 499652), ?, DSJ
- DSJC1000.9.col.b (1000, 898898), ?, DSJ
- DSJC125.1.col.b (125, 1472), ?, DSJ
- DSJC125.5.col.b (125, 7782), ?, DSJ
- DSJC125.9.col.b (125, 13922), ?, DSJ
- DSJC250.1.col.b (250, 6436), ?, DSJ
- DSJC250.5.col.b (250, 31366), ?, DSJ
- DSJC250.9.col.b (250, 55794), ?, DSJ
- DSJC500.1.col.b (500, 24916), ?, DSJ
- DSJC500.5.col.b (500, 125249), ?, DSJ
- DSJC500.9.col.b (500, 224874), ?, DSJ
- DSJR500.1.col.b (500, 7110), ?, DSJ
- DSJR500.1c.col.b (500, 242550), ?, DSJ
- DSJR500.5.col.b (500, 117724), ?, DSJ
UIC DIMACS COLORING BENCHMARKS

- `le450_15b.col` (450,8169), 15, LEI
- `le450_15c.col` (450,16680), 15, LEI
- `le450_15d.col` (450,16750), 15, LEI
- `le450_25a.col` (450,8260), 25, LEI
- `le450_25b.col` (450,8263), 25, LEI
- `le450_25c.col` (450,17343), 25, LEI
- `le450_25d.col` (450,17425), 25, LEI
- `le450_5a.col` (450,5714), 5, LEI
- `le450_5b.col` (450,5734), 5, LEI
- `le450_5c.col` (450,9803), 5, LEI
- `le450_5d.col` (450,9757), 5, LEI
- `mulsol.i.1.col` (197,3925), 49, REG
- `mulsol.i.2.col` (188,3885), 31, REG
- `mulsol.i.3.col` (184,3916), 31, REG
- `mulsol.i.4.col` (185,3946), 31, REG
<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Nodes</th>
<th>Edges</th>
<th>Coloring</th>
<th>SGB/MYC</th>
</tr>
</thead>
<tbody>
<tr>
<td>queen13_13.col</td>
<td>(169,6656)</td>
<td>13</td>
<td>SGB</td>
<td></td>
</tr>
<tr>
<td>queen14_14.col</td>
<td>(196,8372)</td>
<td>?,</td>
<td>SGB</td>
<td></td>
</tr>
<tr>
<td>queen15_15.col</td>
<td>(225,10360)</td>
<td>?,</td>
<td>SGB</td>
<td></td>
</tr>
<tr>
<td>queen16_16.col</td>
<td>(256,12640)</td>
<td>?,</td>
<td>SGB</td>
<td></td>
</tr>
<tr>
<td>queen5_5.col</td>
<td>(25,160)</td>
<td>5</td>
<td>SGB</td>
<td></td>
</tr>
<tr>
<td>queen6_6.col</td>
<td>(36,290)</td>
<td>7</td>
<td>SGB</td>
<td></td>
</tr>
<tr>
<td>queen7_7.col</td>
<td>(49,476)</td>
<td>7</td>
<td>SGB</td>
<td></td>
</tr>
<tr>
<td>queen8_12.col</td>
<td>(96,1368)</td>
<td>12</td>
<td>SGB</td>
<td></td>
</tr>
<tr>
<td>queen8_8.col</td>
<td>(64,728)</td>
<td>9</td>
<td>SGB</td>
<td></td>
</tr>
<tr>
<td>queen9_9.col</td>
<td>(81,2112)</td>
<td>10</td>
<td>SGB</td>
<td></td>
</tr>
<tr>
<td>myciel3.col</td>
<td>(11,20)</td>
<td>4</td>
<td>MYC</td>
<td></td>
</tr>
<tr>
<td>myciel4.col</td>
<td>(23,71)</td>
<td>5</td>
<td>MYC</td>
<td></td>
</tr>
<tr>
<td>myciel5.col</td>
<td>(47,236)</td>
<td>6</td>
<td>MYC</td>
<td></td>
</tr>
<tr>
<td>myciel6.col</td>
<td>(95,755)</td>
<td>7</td>
<td>MYC</td>
<td></td>
</tr>
<tr>
<td>myciel7.col</td>
<td>(191,2360)</td>
<td>8</td>
<td>MYC</td>
<td></td>
</tr>
</tbody>
</table>
DSJ: (From David Johnson (dsj@research.att.com)) Random graphs used in his paper with Aragon, McGeoch, and Schevon, "Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning", Operations Research, 31, 378--406 (1991). DSJC are standard (n,p) random graphs. DSJR are geometric graphs, with DSJR..c being complements of geometric graphs.

CUL: (From Joe Culberson (joe@cs.ualberta.ca)) Quasi-random coloring problem.

REG: (From Gary Lewandowski (gary@cs.wisc.edu)) Problem based on register allocation for variables in real codes.
LEI: (From Craig Morgenstern (morgenst@riogrande.cs.tcu.edu)) Leighton graphs with guaranteed coloring size. A reference is F.T. Leighton, Journal of Research of the National Bureau of Standards, 84: 489--505 (1979).

SCH: (From Gary Lewandowski (lewandow@cs.wisc.edu)) Class scheduling graphs, with and without study halls.

LAT: (From Gary Lewandowski (lewandow@cs.wisc.edu)) Latin square problem.

SGB: (From Michael Trick (trick@cmu.edu)) Graphs from Donald Knuth's Stanford GraphBase

MYC: (From Michael Trick (trick@cmu.edu)) Graphs based on the Mycielski transformation. These graphs are difficult to solve because they are triangle free (clique number 2) but the coloring number increases in problem size
References

- **Krzysztof Wlakowiak**, *Graph coloring using ant algorithms*
- **Gang Qu, Miodrag Potkinjak**, *Analysis of watermarking techniques for graph coloring problem*