Ordering Heuristics for Parallel Graph Coloring

Authors: William Hasenplaugh, Tim Kaler, Tao B. Schardl, Charles E. Leiserson

Presentation by Ethan LaBelle
Definition: Graph-Coloring

- **Definition: Vertex Coloring**
  - Assignment of a color to each vertex of an undirected graph $G = (V, E)$, such that for every edge $(u, v)$ in $E$, $u.color \neq v.color$

- Find optimal vertex coloring (fewest colors)

- NP-complete problem

- In practice, approximation algorithms are sufficient
Motivation

- **Scheduling data graph computations**
  - Sequence of update on vertices of a graph
  - New value of a vertex depends on value of vertex and adjacent vertex values
  - Vertices of same color can be update in parallel
  - Fewer colors $\Leftrightarrow$ more parallelism

- **Other real world applications:**
  - Register allocation via Graph Coloring
Properties of Good Parallel Ordering

- Quality ordering
- Scalable
- Work Efficient
Greedy Algorithm

\text{GREEDY}(G)
1 \textbf{let } G = (V, E, \rho) \\
2 \textbf{for } v \in V \textbf{ in order of decreasing } \rho(v) \\
3 \quad C = \{1, 2, \ldots, \deg(v) + 1\} \\
4 \textbf{for } u \in v.\text{adj such that } \rho(u) > \rho(v) \\
5 \quad C = C - \{u.\text{color}\} \\
6 \quad v.\text{color} = \min C

\(\rho\)- priority function

What is the required work? 
Is this procedure parallelizable?

- Colors a graph with degree \(\Delta\) in at most \(\Delta + 1\) colors
Greedy Algorithm

\textbf{Greedy}(G)
1 \textbf{let} \ G = (V, E, \rho)
2 \textbf{for} v \in V \text{ in order of decreasing } \rho (v)
3 \hspace{1em} C = \{1, 2, \ldots, \deg(v) + 1\}
4 \textbf{for} u \in v.\text{adj} \text{ such that } \rho (u) > \rho (v)
5 \hspace{1em} C = C - \{u.\text{color}\}
6 \hspace{1em} v.\text{color} = \min C

\rho- \text{priority function}

What is the required work?
Is this procedure parallelizable?

- Colors a graph with degree $\Delta$
in at most $\Delta + 1$ colors
Example: Greedy Coloring
Example: Greedy Coloring

Colors
0
1
2
3
4
5

5 4 3 2 1 0

Neighbors Colors
Example: Greedy Coloring
Example: Greedy Coloring

Colors
0 1 2 3 4 5

Neighboring Colors
5 4 3 2 1 0
Example: Greedy Coloring
Example: Greedy Coloring

Colors

Neighboring Colors
Example: Greedy Coloring

Colors
0
1
2
3
4
5

Neighboring Colors

Massachusetts Institute of Technology
Example: Greedy Coloring
Example: Greedy Coloring
Example: Greedy Coloring

Colors
0 1 2 3 4 5

Neighboring Colors
5 4 3 2 1 0
Example: Greedy Coloring
Example: Greedy Coloring

Colors
0
1
2
3
4
5

Neighboring Colors

5 4 3 2 1 0
Example: Greedy Coloring
Example: Greedy Coloring

Colors
0 1 2 3 4 5

Neighboring Colors
5 4 3 2 1 0
Example: Greedy Coloring

The diagram illustrates a graph with nodes colored to represent a greedy coloring.

- **Colors**:
  - Red: 0
  - Blue: 1
  - Green: 2
  - Orange: 3
  - Gray: 4
  - Black: 5

- **Neighboring Colors**

- **Nodes**:
  - 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13

The nodes are colored to minimize the number of colors while ensuring no adjacent nodes have the same color.
Example: Greedy Coloring

The order in which we color the vertices influences the number of colors.

Why so many colors?!?!
Definitions: Ordering Heuristics ($\rho(v)$)

- FF: First fit
- R: Random
- LF: Largest degree first
- SL: Smallest degree last
  - Remove all lowest degree vertices and recursively color graph
- ID: Incidence-degree
  - “Color degree”
- SD: Saturation degree
  - “Distinct color degree”
Example: Largest-First

Colors
0
1
2
3
4
5

Neighboring Colors
5 4 3 2 1 0
Example: Largest-First

Colors

Neighboring Colors

Massachusetts Institute of Technology
Example: Largest-First

Colors
0
1
2
3
4
5

Neighboring Colors
Example: Largest-First

Colors
0 1 2 3 4 5

Neighboring Colors
5 4 3 2 1 0
Example: Largest-First
Example: Largest-First

Colors
0 1 2 3 4 5

Neighboring Colors
Example: Largest-First

Neighboring Colors

5 4 3 2 1 0
Example: Largest-First

Colors
0 1 2 3 4 5

Neighboring Colors
5 4 3 2 1 0
Example: Largest-First

Colors
0
1
2
3
4
5

Neighboring Colors
Example: Largest-First
Example: Largest-First

Colors

0 1 2 3 4 5

Neighboring Colors

5 4 3 2 1 0
Example: Largest-First

Colors

Neighboring Colors
Example: Largest-First

Colors
0 1 2 3 4 5

Neighboring Colors
5 4 3 2 1 0
Example: Largest-First

4 Colors.
Quality vs. Serial Runtime
Quality vs. Serial Runtime
Quality vs. Serial Runtime
Quality vs. Serial Runtime
Parallel Greedy Coloring

\[ \text{JP}(G) \]

7 let \( G = (V, E, \rho) \)
8 \textbf{parallel for } v \in V
9 \hspace{1em} v.\text{pred} = \{u \in V : (u, v) \in E \text{ and } \rho(u) > \rho(v)\}
10 \hspace{1em} v.\text{succ} = \{u \in V : (u, v) \in E \text{ and } \rho(u) < \rho(v)\}
11 \hspace{1em} v.\text{counter} = |v.\text{pred}|
12 \textbf{parallel for } v \in V
13 \hspace{1em} \text{if } v.\text{pred} == \emptyset
14 \hspace{2em} \text{JP-COLOR}(v)

\text{JP-COLOR}(v)

15 \hspace{1em} v.\text{color} = \text{GET-COLOR}(v)
16 \textbf{parallel for } u \in v.\text{succ}
17 \hspace{2em} \text{if JOIN}(u.\text{counter}) == 0
18 \hspace{3em} \text{JP-COLOR}(u)

\text{GET-COLOR}(v)

19 \hspace{1em} C = \{1, 2, \ldots, |v.\text{pred}| + 1\}
20 \textbf{parallel for } u \in v.\text{pred}
21 \hspace{2em} C = C \setminus \{u.\text{color}\}
22 \textbf{return } \min C

Jones and Plassmann [35]

Line 17:
- JOIN(u.\text{counter}) checks if u's predecessors have been colored
Example: Jones-Plassmann

Find root(s) of dag.

Jones and Plassmann - SIAM J. Scientific Computing, 1993
Example: Jones-Plassmann

Jones and Plassmann - SIAM J. Scientific Computing, 1993
Example: Jones-Plassmann

Jones and Plassmann - SIAM J. Scientific Computing, 1993
Example: Jones-Plassmann

Jones and Plassmann - SIAM J. Scientific Computing, 1993
Example: Jones-Plassmann

Jones and Plassmann - SIAM J. Scientific Computing, 1993
Example: Jones-Plassmann

Jones and Plassmann - SIAM J. Scientific Computing, 1993
Example: Jones-Plassmann

Yields the same coloring as the serial Greedy algorithm.

Jones and Plassmann - SIAM J. Scientific Computing, 1993
Analysis

- Linear work in size of the graph
- Traditional heuristics vulnerable to adversarial inputs causing worst case $\Omega(V)$ span
  - Why?
THEOREM 7. For any $\Delta > 0$, there exists a $\Delta$-degree graph $G = (V, E)$ such that JP-LF colors $G$ in $\Omega(\Delta^2)$ span and JP-R colors $G$ in $O(\Delta \lg \Delta + \lg \Delta \Delta \lg V / \lg \lg V)$ expected span.
LLF Ordering Heuristics

- Largest-log-degree-first
- $\rho(v) = \langle \lceil \log(\text{deg}(v)) \rceil, \rho_R(v) \rangle$
- $\rho_R$ is a random priority function
Clique-Chain with JP-LFF
SLL Ordering Heuristic

SLL-ASSIGN-PRIORITIES(G, r)
23  let G = (V, E)
24  i = 1
25  U = V
26  let Δ be the degree of G
27  let \( \rho_R \in \mathbb{R} \) be a random priority function
28  for \( d = 0 \) to \( \log \Delta \)
29    for \( j = 1 \) to \( r \)
30      \( Q = \{ u \in U : |u\text{.adj} \cap U| \leq 2^d \} \)
31      parallel for \( v \in Q \)
32        \( \rho(v) = \langle i, \rho_R(v) \rangle \)
33      \( U = U - Q \)
34    i = i + 1
35  return \( \rho \)
Analysis

- Span bounds:
  - JP-R: $O(\lg V + \lg \Delta \cdot \min \{ \sqrt{E, \Delta} + \lg \Delta \lg V/\lg \lg V \})$
  - JP-LLF: $O(\lg \Delta \lg V + \lg \Delta (\min \{ \sqrt{E, \Delta} + \lg 2\Delta \lg V/\lg \lg V \}))$
  - JP-SLL: $O(\lg \Delta \lg V + \lg \Delta (\min \{ \sqrt{E, \Delta} + \lg 2\Delta \lg V/\lg \lg V \}))$
Empirical Evaluation

Benchmark suite: 8 real-world graphs and 10 synthetic graphs.

<table>
<thead>
<tr>
<th>Serial Heuristic</th>
<th>Parallel Heuristic</th>
<th>Color Ratio</th>
<th>Efficiency</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>R</td>
<td>1.011</td>
<td>0.417</td>
<td>7.039</td>
</tr>
<tr>
<td>LF</td>
<td>LLF</td>
<td>1.021</td>
<td>1.058</td>
<td>7.980</td>
</tr>
<tr>
<td>SL</td>
<td>SLL</td>
<td>1.037</td>
<td>1.092</td>
<td>6.082</td>
</tr>
</tbody>
</table>

**Color Ratio**: Ratio of the number of colors used by the parallel heuristic to the serial heuristic.

**Efficiency**: Ratio of serial heuristic running time to the parallel heuristic run on a single core.

**Speedup**: The 12-core speedup of the parallel heuristic.
“Coarse Hierarchy” In Coloring Quality

FF < R < LLF < LF < SLL < SL
Implementation Techniques

- Join trees for reducing memory contention on atomic counters
  - (Line 17)
- Bit vectors for assigning colors
  - (Line 19) Word containing adjacent colors, maintained during joins
- Software prefetching
  - (Line 16)

```
JP(G)
7   let G = (V,E,ρ)
8   parallel for v ∈ V
9       v.pred = \{ u ∈ V : (u,v) ∈ E and ρ(u) > ρ(v) \}
10      v.succ = \{ u ∈ V : (u,v) ∈ E and ρ(u) < ρ(v) \}
11      v.counter = |v.pred|
12   parallel for v ∈ V
13       if v.pred == ∅
14          JP-COLOR(v)

JP-COLOR(v)
15      v.color = GET-COLOR(v)
16   parallel for u ∈ v.succ
17       if JOIN(u.counter) == 0
18          JP-COLOR(u)

GET-COLOR(v)
19   C = \{1,2,\ldots,|v.pred| + 1\}
20   parallel for u ∈ v.pred
21       C = C \{u.color\}
22   return \min C
```
“Coarse Hierarchy” In Coloring Quality

FF < R < LLF < LF < SLL < SL < SD?
Bonu s: Saturation Degree

\[
\text{THEOREM 13.} \quad \text{GREEDY-SD colors a graph } G = (V, E) \text{ according to the SD ordering heuristic in } \Theta(V + E) \text{ time.}
\]

```
GREEDY-SD(G)
36  let G = (V, E)
37  for v ∈ V
38    v.adjColors = ∅
39    v.adjUncolored = v.adj
40  PUSHORADDKEY(v, Q[0][[v.adjUncolored]])
41  s = 0
42  while s ≥ 0
43    v = POPORDELKEY(Q[s][min KEYS(Q[s])])
44    v.color = min\{1, 2, \ldots, |v.adjUncolored| + 1\} − v.adjColors
45    for u ∈ v.adjUncolored
46      REMOVEORDELKEY(u, Q[|u.adjColors|][|u.adjUncolored|])
47      u.adjColors = u.adjColors ∪ \{v.color\}
48      u.adjUncolored = u.adjUncolored − \{v\}
49      PUSHORADDKEY(u, Q[|u.adjColors|][|u.adjUncolored|])
50    s = max\{s, |u.adjColors|\}
51  while s ≥ 0 and Q[s] ≠ ∅
52     s = s − 1
```
Bonus: Saturation Degree

THEOREM 13. Greedy-SD colors a graph $G = (V, E)$ according to the SD ordering heuristic in $\Theta(V + E)$ time.

- “Saturation Table” Q
- Ordering is determined during serial coloring. How to parallelize?
Acknowledgements

- Professor Leiserson
- Will, Tim
Results

- Overall, JP-LLF obtains a geometric-mean speedup — the ratio of the runtime on 1 core to the runtime on 12 cores — of 7.83 on the eight real-world graphs and 8.08 on the ten synthetic graphs.
- Similarly, JP-SLL obtains a geometric-mean speedup of 5.36 and 7.02 on the real-world and synthetic graphs, respectively.
Incidence Degree

- Iteratively colors an uncolored vertex with the largest number of colored neighbors
Smallest Degree Last

- First remove all lowest degree vertices
- Recursively color the new graph
- Add the removed vertices back and color
Saturation Degree

- Color an uncolored vertex whose colored neighbors use the largest number of distinct colors
Lemma 1

The helper routine GET-COLOR, shown in Figure 2, can be implemented so that during the execution of JP on a graph $G = (V,E,\rho)$, a call to GET-COLOR($v$) for a vertex $v \in V$ costs $\Theta(k)$ work and $\Theta(lgk)$ span, where $k = |v.\text{pred}|$.

Proof:

- Represent set of colors as an array
- Use sentinels to represent removed elements
  - Lines 20-21 require $\Theta(k)$ work and $\Theta(lgk)$ span
- Implement min as a parallel reduction
  - $\Theta(k)$ work and $\Theta(lgk)$ span
- QED
Theorem 2

Given a $\Delta$-degree graph $G = (V,E,\rho)$ for some priority function $\rho$, let $G_\rho$ be the priority dag induced on $G$ by $\rho$, and let $L$ be the depth of $G_\rho$. Then $JP(G)$ runs in $\Theta(V + E)$ work and $O(L\lg\Delta + \lg V)$ span.
Lemma 3

The number of length-\(k\) simple paths in any \(\Delta\)-degree graph \(G = (V,E)\) is at most \(|V| \cdot \min\{\Delta^{k-1}, (2|E|/(k-1))^{k-1}\}\).
Lemma 4

\[ g(\alpha, \beta) = e^2 \frac{\ln \alpha}{\ln \beta} \ln \left( \frac{e^\beta \ln \alpha}{\alpha \ln \beta} \right). \]

Define the function \( g(\alpha, \beta) \) for \( \alpha, \beta > 1 \).

Then for all \( \beta \geq e^2, \alpha \geq 2, \) and \( \beta \geq \alpha \), we have \( g(\alpha, \beta) \geq 1 \).
Theorem 5

Let $G = (V,E)$ be a $\Delta$-degree graph, let $n = |V|$ and $m = |E|$, and let $G_\rho$ be a priority dag induced on $G$ by a random priority function $\rho \in \mathbb{R}$. For any constant $\varepsilon > 0$ and sufficiently large $n$, with probability at most $n^{-\varepsilon}$, there exists a directed path of length $e^2 \cdot \min\{\Delta, \sqrt{m}\} + (1 + \varepsilon)\min\{e^2 \ln \Delta \ln n / \ln \ln n, \ln n\}$ in $G_\rho$. 
Corollary 6. Given a graph $G = (V,E,\rho)$, where $\rho \in \mathbb{R}$ is a random priority function, the expected depth of the priority dag $G_\rho$ is $O(\min\{\sqrt{E, \Delta + \lg \Delta \lg V / \lg \lg V}\})$, and thus JP-R colors all vertices of $G$ with $O(\lg V + \lg \Delta \cdot \min\{\sqrt{E, \Delta + \lg \Delta \lg V / \lg \lg V}\})$ expected span.
Theorem 8

Theorem 8. There exists a class of graphs such that for any $G = (V,E,\rho)$ in the class and for any priority function $\rho \in \text{SL}$, JP-SL incurs $\Omega(V)$ span and JP-R incurs $O(\lg V/\lg \lg V)$ span.
Theorem 9

**Theorem 9.** Let $G = (V,E)$ be a $\Delta$-degree graph, and let $G_\rho$ be the priority dag induced on $G$ by a priority function $\rho \in$ LLF. The expected length of the longest directed path in $G_\rho$ is $O(\min\{\Delta, \sqrt{E}\} + \lg^2\Delta \lg V / \lg \lg V)$. 
Corollary 10

Corollary 10. Given a graph $G = (V, E, \rho)$ for some $\rho \in$ LLF, JP-LLF colors all vertices in $G$ with expected span $O(\lg V + \lg \Delta(\min\{\sqrt{E}, \Delta\} + \lg^2 \Delta \lg V/\lg \lg V))$. □
Corollary 12

**Corollary 12.** Given a graph $G = (V, E, \rho)$ for some $\rho \in \text{SLL}$, JP-SLL colors all vertices in $G$ with expected span $O(\lg \Delta \lg V + \lg \Delta (\min\{\sqrt{E}, \Delta\} + \lg^2 \Delta \lg V / \lg \lg V))$. 
Definition: Vertex-Coloring

- Assignment of a color to each vertex of an undirected graph $G = (V, E)$, such that for every edge $(u, v)$ in $E$, $u.color \neq v.color$