PARALLEL FIRST FIT COLORING

CSE 6490A Winter 2011
Loutfouz Zaman
Overview

• What is graph coloring?
• Applications
• Related work
• Sequential First-Fit
• Parallel First-Fit
• Demo
Graph coloring

• Assignment of "colors" to certain objects in a graph subject to certain constraints
  • Vertex coloring
  • Edge coloring
  • Face coloring (planar)
Vertex coloring

- Coloring vertices of graph such that no two adjacent vertices share same color.
- Edge and Face coloring can be transformed into Vertex version.
- Edge coloring is vertex coloring of its line graph.
Chromatic Number

- $\chi$ - least number of colors needed to color a graph
  - Chromatic number of a complete graph:
    $$\chi(K_n) = n$$
- $\chi(G) = 1$ if and only if $G$ is totally disconnected
- $\chi(G) \leq 4$, for any planar graph
  - The “four-color theorem”
    - More later
Applications of Graph Coloring

- Scheduling
- Register Allocation
- Sudoku
Scheduling (1)

- **Job scheduling**
  - Schedule of interfering jobs
  - Conflict graph
    - Vertices for jobs
    - Edges, if jobs can’t be executed at the same time
    - Colors – time slots

- **Aircraft scheduling**
  - \(k\) aircrafts, \(n\) flights (\(k<n\))
  - 2 flights overlap, same aircraft can’t be used
  - Conflict graph
    - Vertices – flights
    - Edges, if flights overlap
    - Colors - aircrafts
Scheduling (2)

• **Bi-processor tasks**
  • K processors, n tasks
  • Each task has to be executed on pre-assigned processors simultaneously
  • Processor can’t execute 2 jobs at same time
    • E.g. schedule file transfers between processors
    • E.g. mutual diagnostic testing of processors
• Graph
  • Vertices – processors
  • Edge – task between two processors
  • Edge coloring – edge appears at most once at a vertex
Scheduling (3)

- **Frequency assignment**
  - Radio stations at locations marked \((x,y)\)
  - Frequency assigned to each station
    - Interference, must receive different frequencies those that are close
    - E.g. frequency assignment of base stations in cellular phone networks
  - Solved using a 3-approximation algorithm for coloring unit disk graphs
Scheduling (4)

- **Multi-coloring**
  - Earlier example: jobs to have more than one time slots.

- **Pre-coloring extension problem**
  - unassigned vertices using the minimum number of colors

- **List coloring problem**
  - only in certain time slots or machines available
  - Colors are takes from a list of available colors

- **minimum sum coloring**
  - sum of the colors assigned to the vertices is minimal
  - E.g. minimize sum of job completion times -> minimize average completion time
Register Allocation

- Compiler optimization
- Frequently used values are kept in fast processor registers
  - build interference graph ($G$) of program
  - if variables interfere, can’t be assigned to same register
  - Given k register, find $k$-coloring of $G$
  - Uncolored variables are “spilt” into memory

- Recent findings
  - Heuristic approach better allocation than optimal (counter-intuitive) (Koes and Goldstein 2006)
Sudoku (1)

• Fill a 9x9 grid with digits so that each column, each row, and each of the nine 3x3 sub-grids that compose the grid contains all of the digits from 1 to 9

```
<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>6</th>
<th>7</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>8</td>
<td>7</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>3</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>1</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>4</th>
<th>3</th>
<th>5</th>
<th>2</th>
<th>6</th>
<th>9</th>
<th>7</th>
<th>8</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>7</td>
<td>8</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>9</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>2</td>
<td>9</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>9</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>8</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>8</td>
<td>9</td>
<td>5</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>9</td>
</tr>
</tbody>
</table>
```
Sudoku (2)

- Can be viewed graph coloring, here is how:
  - Each one of 81 squares is vertex in graph
  - Edge connects every pair of vertices whose squares are buddies
  - Each vertex connects to 20 other vertices (81x20/2 = 810 edges)
  - Same as to find 9-coloring
  - Also pre-coloring extension problem
Related Work (1)

- Four-Color Theorem
  - Dates back to 1852 to Francis Guthrie
  - Any given plane separated into regions may be colored using no more than 4 colors
    - Used for political boundaries, states, etc
    - Shares common segment (not a point)
  - Many failed proofs, until finally proved using a computer (Appel and Haken 1977)
    - Started in 1972
    - took 1200 hours of computer time
    - Finished 4 years later 😊
Related Work (2)

- Studied as algorithmic problem since early 1970s
- Minimal vertex coloring algorithm using brute-force search
  - Christodes 1971
  - Wilf 1984
- Finding minimum coloring: \textbf{NP}-hard
  - You can’t do it efficiently for large graphs
- Approximations
  - guarantee performance at expense of quality
    - quality = \# colors used
  - E.g. Brelaz 1979
    - Good but not minimal solution
    - Minimal for certain type of graphs
Related Work (3)

• State of the art
  • Pushing tradeoff limits between performance and used number of colors

• Schneider and Wattenhofer 2010
  • Algorithm for distributed symmetry breaking

<table>
<thead>
<tr>
<th>Colors</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Delta + 1)</td>
<td>(O(\log \Delta + \sqrt{\log n}))</td>
</tr>
<tr>
<td>(O(\Delta + \log n))</td>
<td>(O(\log \log n))</td>
</tr>
<tr>
<td>(O(\Delta + \log^{1+1/\log^* n} n))</td>
<td>(O(\log^* n))</td>
</tr>
</tbody>
</table>
| \(O(\Delta \log^{(c)} n + \log^{1+1/c} n)\) | \(O(1)\)
Related Work (4)

- Online coloring
  - Approximation
  - Heuristic algorithms used to produce proper graph coloring which is not necessarily minimal
  - Immediately colors vertices of $G$ taken from list without looking ahead or changing colors already assigned
  - Any online algorithm lower bounds (Halldórsson and Szegedy 1994):

<table>
<thead>
<tr>
<th>Deterministic</th>
<th>Randomized</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\geq 0 \left( \frac{2n}{\log^{n^2}} \right)$</td>
<td>$\geq 0 \left( \frac{n}{16\log^{n^2}} \right)$</td>
</tr>
</tbody>
</table>
Related Work (5)

- First-Fit (FF) simplest of all online coloring algorithms
- Assigns smallest possible integer as color to current vertex of $G$ (Gyárfás and Lehel 1988)
- Appears extensively with *interval graphs*
  - Interval graph captures intersection relation for some set of intervals on real line
    - E.g. resource requests arrive dynamically in unpredictable order
    - FF allocates lowest color to current interval that respects constraints imposed by colored intervals
Related Work (6)

• How bad is FF compared to optimal coloring?
  • $\chi_{FF}(G)$ – maximum number of colors used for colorings of $G$ produced by FF for all orderings of vertices of $G$
  • $\chi(G)$ – chromatic number of $G$
  • Performance ratio of FF: $R_{FF} = \chi_{FF}(G)/\chi(G)$
  • Recent findings: $5 \leq \chi_{FF}(G) \leq 8$
  • Wan et al. (2010) used FF for First-Fit scheduling as an approximation algorithm for minimum latency beaconing schedule
Sequential FF (1)

• Umland (1998) demonstrates a 2-step sequential FF algorithm:
  
  (1) **Build**($L_i, v_j$): Determine a list $L_i$ of all possible colors for $v_i$, i.e. exclude colors already used by vertices $v_j, j < i$ adjacent to $v_i$

    - $L_i$ -- a boolean array (possibility list of $v_i$) with property:
      - $L_i[k] = false \leftrightarrow \exists v_j$ such that $j < i, (v_i, v_j) \in E$ and $f(v_j) = k$

  (2) **Color**($L_i, v_i$): Determine the smallest of all possible colors for $v_i$, i.e. look for the smallest entry in $L_i$ where $L_i[k] = true$ and assign color $k$ to $v_i$
Sequential FF (2)

Algorithm 1 Build($L_i, v_j$)

Require: must be executed before $\text{Color}(L_i, v_j), \forall j < i$ and requires $L_i$ initialized

Ensure: $L_i[k] = \text{false} \iff \exists v_j$ such that $j < i, (v_i, v_j) \in E, f(v_j) = k$

1: for all $n$ in $v_i.\text{neighbours}$ do
2:     if $n.\text{index} > v_i.\text{index}$ then
3:         $L_{n.\text{index}}[v_i.\text{color}] \leftarrow \text{false}$
4:     end if
5: end for

Algorithm 2 Color($L_i, v_i$)

Require: must be executed before $\text{Build}(L_j, v_i), \forall j > i$

Ensure: $v_i$ has the first color unused by neighbours

1: for $k = 0$ to $L_i.\text{length}$ do
2:     if $L_i[k] = \text{true}$ then
3:         $v_i.\text{color} \leftarrow k$
4:         break
5:     end if
6: end for
Sequential FF E.g. Step 0
Sequential FF E.g. Step 1

$L_1 = \{t, t, t\}, k=0$

$L_4 = \{f, t, t, t\}$  $L_6 = \{f, t, t, t\}$  $L_6 = \{f, t, t, t\}$
Sequential FF E.g. Step 2

$L_1 = \{t, t, t\}, k=0$
$L_3 = \{f, t, t, t\}$
$L_5 = \{f, t, t, t\}$
$L_7 = \{f, t, t, t\}$

$L_2 = \{t, t, t\}, k=0$
$L_4 = \{f, t, t, t\}$
$L_6 = \{f, t, t, t\}$
$L_8 = \{f, t, t, t\}$
Sequential FF E.g. Step 3

$L_1 = \{t, t, t\}, k=0$
$L_2 = \{t, t, t\}, k=0$

$L_3 = \{f, t, t, t\}, k = 1$
$L_4 = \{f, t, t, t\}$

$L_5 = \{f, t, t, t\}$
$L_6 = \{f, f, t, t\}$

$L_7 = \{f, t, t, t\}$
$L_8 = \{f, f, t, t\}$
Sequential FF E.g. Step 4

$L_1 = \{t, t, t\}, k = 0$
$L_3 = \{f, t, t, t\}, k = 1$
$L_5 = \{f, f, t, t\}$
$L_7 = \{f, f, t, t\}$

$L_2 = \{t, t, t\}, k = 0$
$L_4 = \{f, t, t, t\}, k = 1$
$L_6 = \{f, f, t, t\}$
$L_6 = \{f, f, t, t\}$
Sequential FF E.g. Step 5

\[
L_1 = \{t, t, t\}, k=0 \quad L_3 = \{f, t, t, t\}, k=1 \quad L_5 = \{f, f, t, t\}, k=2 \quad L_7 = \{f, f, t, t\}
\]

\[
L_2 = \{t, t, t\}, k=0 \quad L_4 = \{f, t, t, t\}, k=1 \quad L_6 = \{f, f, t, t\} \quad L_6 = \{f, f, f, t\}
\]
Sequential FF E.g. Step 6

\[ L_1 = \{ t, t, t \}, k = 0 \]
\[ L_2 = \{ t, t, t \}, k = 0 \]
\[ L_3 = \{ f, t, t, t \}, k = 1 \]
\[ L_4 = \{ f, t, t \}, k = 1 \]
\[ L_5 = \{ f, f, t, t \}, k = 2 \]
\[ L_6 = \{ f, f, t, t \}, k = 2 \]
\[ L_7 = \{ f, f, f, t \} \]
\[ L_8 = \{ f, f, f, t \} \]
Sequential FF E.g. Step 7

$L_1 = \{t, t, t, t\}, k=0$

$L_2 = \{t, t, t, t\}, k=0$

$L_3 = \{f, t, t, t\}, k=1$

$L_4 = \{f, t, t, t\}, k=1$

$L_5 = \{f, f, t, t\}, k=2$

$L_6 = \{f, f, t, t\}, k=2$

$L_7 = \{f, f, f, t\}, k=3$

$L_8 = \{f, f, f, f\}$
Sequential FF E.g. Step 8

\[ L_1 = \{t, t, t\}, k = 0 \]
\[ L_2 = \{t, t, t\}, k = 0 \]
\[ L_3 = \{f, t, t, t\}, k = 1 \]
\[ L_4 = \{f, t, t\}, k = 1 \]
\[ L_5 = \{f, f, t, t\}, k = 2 \]
\[ L_6 = \{f, f, t, t\}, k = 2 \]
\[ L_7 = \{f, f, f, t\}, k = 3 \]
\[ L_8 = \{f, f, f, f\}, k = 4 \]
Parallel FF (1)

Step   Processor\textsubscript{1}   Processor\textsubscript{2}   Processor\textsubscript{3}   Processor\textsubscript{4}   Processor\textsubscript{5}

1.     Color($L_1, v_1$)

2.     Build($L_2, v_1$)

3.     Build($L_3, v_1$)   Color($L_2, v_2$)

4.     Build($L_4, v_1$)   Build($L_3, v_2$)

5.     Build($L_5, v_1$)   Build($L_4, v_2$)   Color($L_3, v_3$)

6.     Build($L_5, v_2$)   Build($L_4, v_3$)

7.     Build($L_5, v_3$)   Color($L_4, v_4$)

8.     Build($L_5, v_4$)

9.     Color($L_5, v_5$)

Figure 2: Parallel first fit with 5 vertices and 5 processors.
Parallel FF (2)

- Problem
  - Requires same number of cores as there are vertices in $G$

- Generalized algorithm
  - Processors $P_1, \ldots, P_n (1 \leq N \leq n), n – vertices$
  - Every processor colors whole subgraph with $n/N$ instead of single vertex unlike
  - Possibility lists prepared on previous processors
  - $Build(L_i, V_j)$ excludes colors of all vertices
  - $V_j = \{v_{1+(j-1)n/N}, \ldots, v_{jn/N}\}$ in $j^{th}$ subgraph from $L_i$ which will be
  - later by another processor
Generalized Parallel FF (1)

Figure 3: Generalized parallel first fit (16 vertices, 4 processors).
Generalized FF (2)

- Roughly 50% of resources not used
  - Speedup is not expected to exceed half the number of cores
  - Still good for this type of algorithm

- Implementation
  - Share graph among cores
  - Flow of control over $L_t$ (illustrated by arrows) can be implemented by passing tokens from thread to thread
    - CSP
    - No need to transfer entire list
Umland’s Results

Parallel graph coloring (graph with 2000 vertices)
Plan(1)

- Umland ran algorithm on SPARC 40 MHz and 128 MB RAM in 1998
  - 2000 vertices, 999001 edges
  - 1001 colors
  - 30 seconds
- Implement generalized parallel algorithm
- Determine graph of comparable size for modern hardware
- Run with different number of threads and observe speedup or improvements in total time
- DEMO TIME!!
Questions?