Graph Coloring in Parallel Processing and Scientific Computing

Assefaw Gebremedhin

CSCAPES Institute and
Old Dominion University
assefaw@cs.odu.edu
www.cs.odu.edu/~assefaw

Joint work with
Alex Pothen (Purdue)
Doruk Bozdag and Umit Catalyurek (Ohio State Univ)
Erik Boman (Sandia)
Fredrik Manne (Univ of Bergen)
Andrea Walther (Tech Univ of Dresden)
Arijit Tarafdar and Duc Nguyen (ODU)

CSCAPES Workshop, June 2008, Santa Fe, NM
A distance-1 coloring of $G = (V, E)$ is
- a mapping $\phi : V \rightarrow \{1, 2, \ldots, q\}$ s.t.
  $\phi(u) \neq \phi(v)$ whenever $(u, v) \in E$
- a partitioning of $V$ into $q$ independent sets

The objective is to minimize $q$

Distance-1 coloring is used to discover concurrency in parallel scientific computing. Examples:
- iterative methods for sparse linear systems (Jones & Plassmann, 94)
- adaptive mesh refinement
- preconditioners (Saad, 96; Hysom & Pothen, 01)
- eigenvalue computation (Manne, 98)
- sparse tiling (Strout et al, 02)
**Procedure** \( \text{SPARSECOMPUTE}(F : \mathbb{R}^n \rightarrow \mathbb{R}^m) \)

**S1.** Determine the sparsity structure of the derivative (first or second) matrix \( A \in \mathbb{R}^{m \times n} \) of the function \( F \)

**S2.** Obtain a seed matrix \( S \in \{0, 1\}^{n \times q} \) with the smallest \( q \)

**S3.** Compute the numerical values of the entries of the compressed matrix \( B = AS \in \mathbb{R}^{m \times q} \)

**S4.** Recover the numerical values of the entries of \( A \) from \( B \)

The seed matrix \( S \) partitions the columns of \( A \):

\[
s_{jk} = \begin{cases} 
1 & \text{iff column } a_j \text{ belongs to group } k, \\
0 & \text{otherwise.}
\end{cases}
\]

It is obtained using an appropriate coloring on the graph of \( A \).
COLORING MODEL VARIATIONS IN DERIVATIVE COMPUTATION VIA COMPRESSION

Sources of problem variation:

- **Type of derivative matrix**
  - Jacobian (nonsymmetric)
  - Hessian (symmetric)

- **Recovery method**
  - Direct
  - Substitution

- **Dimension of partitioning** (for the Jacobian case)
  - Unidirectional (only columns or rows)
  - Bidirectional (both columns and rows)
An archetypal model for direct methods

Structurally orthogonal partition of matrix $A$ equivalent to:

- **Distance-2 coloring** of the adjacency graph $G_a(A) = (V, E)$ when $A$ is symmetric (McCormick, 1983)

- **Partial distance-2 coloring** of the bipartite graph $G_b(A) = (V_1, V_2, E)$ when $A$ is nonsymmetric (GMP, 2005)

- **Distance-1 coloring** of the appropriate square graph (Coleman and Moré, 1983)
**Symmetrically orthogonal partition**: whenever $h_{ij} \neq 0$

- $h_j$ only column in a group with nonzero at row $i$ or
- $h_i$ only column in a group with nonzero at row $j$

**Star coloring**: a vertex coloring $\phi$ of $G_a(H)$ s.t.

- $\phi$ is a distance-1 coloring and
- every path on 4 vertices ($P_4$) uses at least 3 colors

**SymOP equivalent to star coloring** (Coleman and Moré, 84)
AN ACCURATE MODEL FOR HESSIAN COMPUTATION VIA SUBSTITUTION

Substitutable partition: whenever $h_{ij} \neq 0$
- $h_j$ in a group where all nonzeros in row $i$ are ordered before $h_{ij}$ or
- $h_i$ in a group where all nonzeros in row $j$ are ordered before $h_{ij}$

Acyclic coloring: a vertex coloring $\phi$ of $G_a(H)$ s.t.
- $\phi$ is a distance-1 coloring and
- every cycle uses at least 3 colors

Substitutable partition equivalent to acyclic coloring (Coleman and Cai, 86)
Overview of coloring models in derivative computation

### General sparsity pattern:

<table>
<thead>
<tr>
<th></th>
<th>unidirectional partition</th>
<th>bidirectional partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacobian</td>
<td>distance-2 coloring</td>
<td>star bicoloring</td>
</tr>
<tr>
<td>Hessian</td>
<td>star coloring</td>
<td>NA</td>
</tr>
<tr>
<td>(restricted star coloring)</td>
<td></td>
<td>direct bicoloring</td>
</tr>
<tr>
<td>Jacobian</td>
<td>NA</td>
<td>acyclic bicoloring</td>
</tr>
<tr>
<td>Hessian</td>
<td>acyclic coloring</td>
<td>NA</td>
</tr>
<tr>
<td>(triangular coloring)</td>
<td></td>
<td>substitution bicoloring</td>
</tr>
</tbody>
</table>

Nonsym $A$ \quad $G_b(A) = (V_1, V_2, E)$

Sym $A$ \quad $G(A) = (V, E)$

### Regular sparsity pattern (discretization of structured grids):

- **Formula-based coloring** (Goldfarb and Toint, 1984)
- **Hierarchical coloring** (Hovland, 2007)
1. **Models**
   - Parallel scientific computing
   - Derivative computation

2. **Sequential algorithms**

3. **Case studies**

4. **Parallel algorithms**

5. **Summary**
Distance-\(k\), star, and acyclic coloring are NP-hard (they are also hard to approximate)

A greedy heuristic usually gives a good solution

\[ \text{GREEDY}(G = (V, E)) \]

Let \( v_1, v_2, \ldots, v_n \) be an ordering of \( V \)

\begin{align*}
\text{for } i = 1 \text{ to } n \text{ do} & \\
& \text{determine forbidden colors to } v_i \\
& \text{assign } v_i \text{ the smallest permissible color}
\end{align*}

\[ \text{end-for} \]

For distance-\(k\) coloring, \text{GREEDY} can be implemented to run in \( O(n\bar{d}_k) \) time, where \( \bar{d}_k \) is the average degree-\(k\)

We have developed \( O(n\bar{d}_2) \)-time heuristic algorithms for star and acyclic coloring

\textbf{Key idea:} exploit the structure of two-colored induced subgraphs
Algorithm (Input: $G = (V, E)$):  

for each $v \in V$  

1. Choose color for $v$  
   - forbid colors used by neighbors $N(v)$ of $v$  
   - forbid colors leading to two-colored $P_4$  
     - $\forall \{w, x\} \subseteq N(v)$ where $\phi(w) = \phi(x)$, forbid colors used by $N(w)$ and $N(x)$  
     - $\forall$ non-single-edge star $S$ incident on $v$, forbid color of hub of $S$  

2. Update collection of two-colored stars  

Time: $O(|V|d_2)$  
Space: $O(|E|)$
Algorithm (Input: $G = (V, E)$):
for each $v \in V$

1. Choose color for $v$
   - forbid colors used by neighbors $N(v)$ of $v$
   - forbid colors leading to two-colored cycles
     - $\forall$ tree $T$ incident on $v$, if $v$ adj to $\geq 2$ vertices of same color, forbid the other color in $T$

2. Update collection of two-colored trees (merge if necessary)

Time: $O(|V| \bar{d} \cdot \alpha)$  
Space: $O(|E|)$
Performance Comparison:

New Star and Acyclic Coloring Algorithms vs Previous Algorithms

<table>
<thead>
<tr>
<th>V in 1000</th>
<th>E in 1000</th>
<th>MaxDeg</th>
<th>MinDeg</th>
<th>AvgDeg</th>
</tr>
</thead>
<tbody>
<tr>
<td>range</td>
<td>10 – 150</td>
<td>50 – 17,000</td>
<td>8 – 860</td>
<td>0 – 230</td>
</tr>
<tr>
<td>sum</td>
<td>1,500</td>
<td>88,000</td>
<td>6,400</td>
<td>800</td>
</tr>
</tbody>
</table>

Table: Summary of size and density of test graphs (total: 29).

<table>
<thead>
<tr>
<th></th>
<th>D2</th>
<th>RS</th>
<th>NS</th>
<th>S</th>
<th>T-sl</th>
<th>A</th>
<th>D1</th>
</tr>
</thead>
<tbody>
<tr>
<td>colors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>9,240</td>
<td>8,749</td>
<td>7,636</td>
<td>7,558</td>
<td>5,065</td>
<td>4,110</td>
<td>1,757</td>
</tr>
<tr>
<td></td>
<td>28.2</td>
<td>34.4</td>
<td>930</td>
<td>162</td>
<td>12.4</td>
<td>32.5</td>
<td>0.04</td>
</tr>
</tbody>
</table>

Table: Total number of colors and runtime, summed over all test cases.
Outline

1 Models
- Parallel scientific computing
- Derivative computation

2 Sequential algorithms

3 Case studies

4 Parallel algorithms

5 Summary
Experiments using ADOL-C

- Efficacy of the four-step scheme tested in two case studies
  1. Jacobian computation in a Simulated Moving Bed process (chromatographic separation in chemical engineering)
  2. Hessian computation in an optimal electric power flow problem
- Experiments showed
  - technique enabled cheap Jacobian/Hessian computation where dense computation is infeasible
  - observed results for each step matched analytical results
1 Models
   - Parallel scientific computing
   - Derivative computation

2 Sequential algorithms

3 Case studies

4 Parallel algorithms

5 Summary
Parallelizing greedy coloring

- Desired task: parallelize **Greedy** such that
  - speedup is $\Theta(p)$
  - number of colors used is roughly same as in serial
- A difficult task since **Greedy** is inherently sequential
- For D1 coloring, several approaches based on Luby’s parallel algorithm for maximal independent set exist
- Some drawbacks:
  - no actual parallel implementation
  - many more colors than a serial implementation
  - poor parallel speedup on unstructured graphs
Generic parallelization techniques

- **Basic standard techniques:**
  - balanced trees, pointer jumping,
  - divide and conquer, strict partitioning

- **Strict partitioning:**
  - break up the given problem into \( p \) independent subproblems of almost equal sizes
  - solve the \( p \) subproblems concurrently using \( p \) processors

Main work in SP lies in the decomposition step, often no easier than solving the original problem.

- **Relaxed partitioning:**
  - break up the given problem into \( p \), not necessarily entirely independent, subproblems of almost equal sizes
  - solve the \( p \) subproblems concurrently
  - detect inconsistencies in the solutions concurrently
  - resolve any inconsistencies

RP can be used successfully if the resolution in the fourth step involves only “local” adjustments.
RP applied to greedy coloring

Basic features of the algorithm:

- exploits features of data distribution
  - distinguishes between interior and boundary vertices
- proceeds in rounds, each having two phases:
  - tentative coloring
  - conflict detection
- tentative coloring phase organized in supersteps
  - each processor communicates only after coloring a subset of its assigned vertices using currently available information (infrequent, coarse-grain communication)
- randomization used in resolving conflicts
A Framework for Parallel Distance-1 Coloring

**Framework** \((G = (V, E), s)\)

**Partition** \(V\) into \(V_1, V_2, \ldots, V_p\) using a graph partitioner

**On each processor** \(P_i, i \in I = \{1, \ldots, p\}\)

- for each boundary vtx \(v \in V'_i = \{u : (u, v) \in E_i\}\)
  - assign \(v\) a random number \(r(v)\)
  - \(U_i \leftarrow V_i\)

while \(\exists j \in I, U_j \neq \emptyset\) rounds

- Partition \(U_i\) into \(\ell_i\) subsets \(U_{i,1}, U_{i,2}, \ldots, U_{i,\ell_i}\), each of size \(s\)
- for \(k = 1\) to \(\ell_i\) do supersteps for tentative coloring
  - for each \(v \in U_{i,k}\) do
    - assign \(v\) a permissible color
    - send colors of boundary vtxs in \(U_{i,k}\) to relevant processors
    - receive color information from relevant processors

Wait until all incoming messages are received

- \(R_i \leftarrow \emptyset\)
- for each boundary vtx \(v \in U_i\) do conflict detection
  - if \(\exists (v, w) \in E_i\) s.t. \(c(v) = c(w)\) and \(r(v) < r(w)\) then
    - \(R_i \leftarrow R_i \cup \{v\}\)

- \(U_i \leftarrow R_i\) recolor in next round
Specializations of Framework

Framework can be specialized along several axes:

1. **Color selection strategies:**
   - **First Fit:** search for smallest color starts at 1 on each processor
   - **Staggered FF:** search for smallest color starts from different “bases”

2. **Coloring order:**
   - interior vertices can be colored \textit{before, after, or interleaved with} boundary vertices

3. **Local vertex ordering:**
   - vertices on each processor can be ordered using various degree-based techniques

4. **Supersteps:**
   - can be run \textit{synchronously} or \textit{asynchronously}

5. **Inter-processor communication:**
   - can be \textit{customized} or \textit{broadcast-based}
How should the **options** in Framework be set?

An answer requires considering a complex set of factors, including:

- size and density of input graph
- number of processors
- quality of initial partitioning
- characteristic of platform on which implementation is run

Determination bound to rely on experimentation
Lessons learned from experiments

Good parameter configuration for large-size (millions of edges) graphs:

- **moderately unstructured** graphs (e.g. a typical application graph):
  - a superstep size $s$ in the order of 1000
  - asynchronous supersteps
  - a coloring order in which interior vertices appear either strictly before or strictly after boundary vertices
  - First Fit color choice strategy
  - customized inter-processor communication

- **highly unstructured** (e.g. random) graphs:
  - $s$ in the order of 100
  - items 2 to 4 same as for moderately unstructured graphs
  - broadcast-based communication
Algorithm FBAC on Itanium 2 cluster.
A sample experimental result: weak scalability

Itanium 2

Pentium 4

Assefaw Gebremedhin (CSCAPES)  Coloring in Scientific Computing
SUMMARY

Current accomplishments:

- Developed a unifying graph-theoretic framework for sparse derivative computation.
- Designed and implemented new sequential algorithms for distance-\( k \), star, acyclic, and other coloring problems.
- C++ implementations assembled in a package called ColPack.
  - ColPack also includes various ordering routines for greedy coloring.
- Integrated parts of ColPack with the AD tool ADOL-C.
- Developed parallel algorithms for distance-1, distance-2, and restricted star coloring.
  - Algorithms scale well for a hundred processors.
  - Implementations made available via Zoltan.

Planned activities:

- Integrate coloring software with tools in OpenAD.
- Develop algorithms for coloring problems in partial matrix computation.
- Develop parallel star and acyclic coloring algorithms.
- Develop parallel coloring algorithms for tera and petascale computation.
- Collaborate with application and tool developers to “plug in” coloring technologies to enable CSE.
Gebremedhin, Manne and Pothen.
What Color Is Your Jacobian? Graph Coloring for Computing Derivatives. 

Gebremedhin, Tarafdar, Manne and Pothen.
New Acyclic and Star Coloring Algorithms with Application to Computing Hessians. 

Gebremedhin, Pothen and Walther.
Exploiting Sparsity in Jacobian Computation via Coloring and Automatic Differentiation: 
A Case Study in a Simulated Moving Bed Process. 

Gebremedhin, Pothen, Tarafdar and Walther.
Efficient Computation of Sparse Hessians using Coloring and Automatic Differentiation. 

Bozdag, Gebremedhin, Manne, Boman and Catalyurek.
A Framework for Scalable Greedy Coloring on Distributed-memory Parallel Computers. 

Bozdag, Catalyurek, Gebremedhin, Manne, Boman and Ozguner.
Distributed-memory Parallel Graph Coloring Algorithms for Jacobian and Hessian Computation. 