Graph Coloring in Scientific Computing
An example of the enabling power of combinatorial algorithms in scientific computing

Assefaw Gebremedhin
Purdue University
www.cs.purdue.edu/homes/agebreme
agebreme@purdue.edu

Dagstuhl Seminar on Combinatorial Scientific Computing
Feb 1—6, 2009
Joint work with…

- Alex Pothen (Purdue)
- Fredrik Manne (Bergen)
- Andrea Walther (Dresden)
- Duc Nguyen and Arijit Tarafdar (ODU)
- Doruk Bozdag and Umit Catalyurek (Ohio State)
- Erik Boman (Sandia)

Financially supported by…

- Department of Energy
- National Science Foundation
Combinatorial Scientific Computing:
The use of combinatorial algorithms (along two orthogonal fronts) in the context of solving mathematical problems in science and engineering using computers

Front 1: domain
- Traditional scientific computing
  - numerical linear algebra
  - numerical solution of differential equations
  - numerical optimization
- Computational science and engineering
- Informatics

Front 2: infrastructural technologies for HPC
- load balancing
- task scheduling
- data migration
- topology-aware mapping
- improving data locality in irregular computation
- etc
Unifying features of CSC problems and scenarios

- Overarching goal is to make computation efficient or feasible
- Accurate abstraction challenging to find
- Abstractions often expressed in terms of graphs or hypergraphs
- Identified problems often NP-hard; fast approx algorithms needed
- Algorithms need to be parallelized, to avoid being bottlenecks
- Software development and deployment of paramount importance
Focus of this talk
Outline

1. Models
2. Sequential algorithms
   ◦ Software
3. A case study
4. Parallel algorithms
   ◦ Software
**Coloring in automatic differentiation: context**

**Procedure** \( \text{SPARSECOMPUTE}( F : \mathbb{R}^n \rightarrow \mathbb{R}^m ) \)

1. Determine the \textbf{sparsity structure} of \( F' = A \in \mathbb{R}^{m \times n} \)
2. Obtain a \textbf{seed} matrix \( S \in \{0,1\}^{n \times q} \) of the smallest \( q \)
3. Compute the \textbf{compressed} matrix \( B = AS \in \mathbb{R}^{m \times q} \)
4. \textbf{Recover} the numerical values of the entries of \( A \) from \( B \)

Seed matrix \( S \) \textbf{partitions} columns of \( A \):

\[
S_{jk} = \begin{cases} 
1 & \text{iff column } j \text{ of } A \text{ belongs to group } k \\
0 & \text{otherwise}
\end{cases}
\]

\( S \) obtained using an appropriate \textbf{coloring} of the graph of \( A \)
Sources of model variation in derivative computation via compression

Three orthogonal axes, each with two possibilities:

<table>
<thead>
<tr>
<th>Type of Derivative Matrix</th>
<th>Recovery Method</th>
<th>Dimension of Partitioning*</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Jacobian (nonsymmetric)</td>
<td>• Direct</td>
<td>• Unidirectional</td>
</tr>
<tr>
<td>• Hessian (symmetric)</td>
<td>• Substitution</td>
<td>• Bidirectional</td>
</tr>
</tbody>
</table>

* for the Jacobian case only
Distance-2 coloring: an archetypal model in direct methods

**Symmetric case**

A

\[
\begin{bmatrix}
  a_{11} & 0 & 0 & 0 & a_{15} \\
  0 & a_{22} & a_{23} & 0 & 0 \\
  0 & a_{32} & a_{33} & a_{34} & a_{35} \\
  0 & 0 & a_{43} & a_{44} & a_{45} \\
  a_{51} & 0 & a_{53} & a_{54} & a_{55}
\end{bmatrix}
\]

**Distance-2 coloring**

**Distance-1 coloring**

**Non-symmetric case**

Curtis, Powell and Reid, 74

\[
\begin{bmatrix}
  a_{11} & a_{12} & 0 & 0 & a_{15} \\
  a_{21} & a_{22} & 0 & 0 & 0 \\
  a_{31} & 0 & 0 & a_{34} & 0 \\
  0 & 0 & a_{43} & a_{44} & a_{45}
\end{bmatrix}
\]

McCormick, 83

Coleman and More, 83
An accurate model for Hessian computation via a direct method

symmetrically orthogonal partition

star coloring:
- distance-1 coloring +
- every path on 4 vertices uses at least 3 colors

Model due to Coleman and More (84);
A less accurate model, equivalent to restricted star coloring, was suggested by Powell and Toint in 79.

\[
B = HS
\]
An accurate model for Hessian computation via substitution

substitutable partition

acyclic coloring:
• distance-1 coloring +
• every cycle uses at least 3 colors

Model due to Coleman and Cai (86);
A less accurate model, called triangular coloring, was suggested by Coleman and More in 84.

\[
\begin{pmatrix}
h_1 & h_{12} + h_{17} & 0 \\
h_{21} + h_{23} + h_{25} & h_2 & 0 \\
h_33 & h_{32} + h_{34} & h_{36} \\
h_{43} + h_{4,10} & h_{44} & 0 \\
h_{55} & h_{52} & h_{56} + h_{58} \\
h_{63} + h_{65} & h_{69} & h_{66} \\
h_{71} & h_{77} & h_{78} \\
h_{85} & h_{87} + h_{89} & h_{88} \\
h_{9,10} & h_{99} & h_{96} + h_{98} \\
h_{10,10} & h_{10,4} + h_{10,9} & 0 \\
\end{pmatrix}
\]

compressed Hessian
\[ B = HS \]
Bidirectional Jacobian computation

- **Direct method:**
  - star bicoloring (shown in the picture)
    - (Coleman & Verma’98 and Hossain & Steihaug’98)

- **Substitution method:**
  - acyclic bicoloring (Coleman & Verma’98)
## Summary 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><strong>Jacobian</strong></td>
<td>distance-2 coloring</td>
<td>star bicoloring</td>
</tr>
<tr>
<td><strong>Hessian</strong></td>
<td>star coloring</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>restricted star coloring</td>
<td></td>
</tr>
<tr>
<td><strong>Jacobian</strong></td>
<td>NA</td>
<td>acyclic bicoloring</td>
</tr>
<tr>
<td><strong>Hessian</strong></td>
<td>acyclic coloring</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>triangular coloring</td>
<td></td>
</tr>
</tbody>
</table>

### Regular sparsity pattern:

- Formula-based coloring (Goldfarb & Toint’84)
- Hierarchical coloring (Hovland’07)

### Further reading:


---

Jacobian: bipartite graph
Hessian: adjacency graph
Outline

1. Models
2. Sequential algorithms
   - Software
3. A Case study
4. Parallel algorithms
   - Software
Complexity and algorithms

- Distance-k, star, and acyclic coloring are NP-hard
  - also hard to approximate
- A greedy heuristic usually gives good solution

\[
\text{GREEDY}(G=(V,E))
\]
\begin{algorithm}
    \textbf{Order} the vertices in \( V \)
    \begin{algorithmic}
        \For {i = 1 \text { to } |V|} 
            \State Determine \textit{forbidden} colors to \( v_i \)
            \State Assign \( v_i \) the \textit{smallest} permissible color
        \EndFor
    \end{algorithmic}
\end{algorithm}

- For distance-k coloring, \text{GREEDY} can be implemented to run in \( O(|V|d_k) \) time, where \( d_k \) is average degree-k
- We have developed \( O(|V|d_2) \)-time heuristic algorithms for star and acyclic coloring

\textit{Key idea}: exploit structure of two-colored induced subgraphs
### A new star coloring algorithm

**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 paths on 4 vertices:  
     - For every pair of same-colored vertices $w$ and $x$ in $N(v)$, forbid colors used by $N(w)$ and $N(x)$
     - For every 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:** $O(|E|)$
A new acyclic coloring algorithm

Algorithm (Input: $G=(V,E)$):

for each $v$ in $V$
1. Choose color for $v$
   • Forbid colors used by neighbors of $v$
   • Forbid colors leading to two-colored cycles
     • For every tree $T$ incident on $v$, if $v$ adjacent to at least two same-color vertices, forbid the other color in $T$
2. Update collection of two-colored trees (merge if necessary)

Time: $O(|V|d_2 \alpha)$  Space: $O(|E|)$
Performance

Number of colors

- D2: 9240 colors
- RS: 8749 colors
- S: 7558 colors
- T-sl: 5065 colors
- A: 4110 colors
- D1: 1757 colors

Runtime (min)

- D2: 28.2 minutes
- RS: 34.4 minutes
- S: 12.4 minutes
- T-sl: 32.5 minutes
- A: 0.04 minutes
- D1: 0.04 minutes

29 test graphs, aggregate data

<table>
<thead>
<tr>
<th></th>
<th>V</th>
<th></th>
<th>E</th>
<th>Max Degree</th>
<th>Min Degree</th>
<th>Avg Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.5M</td>
<td>88M</td>
<td>6.4K</td>
<td>800</td>
<td>4.2K</td>
<td></td>
</tr>
</tbody>
</table>

Further reading:
Hessian recovery algorithms

Direct (star coloring)

\[
\begin{bmatrix}
  h_{11} & h_{12} & h_{17} & 0 & 0 \\
  h_{21} + h_{23} + h_{25} & h_{22} & 0 & 0 & 0 \\
  h_{31} & h_{32} & h_{34} & h_{36} & 0 \\
  h_{41} & h_{410} & h_{41} & 0 & 0 \\
  h_{55} & h_{52} & 0 & h_{56} & h_{58} \\
  h_{65} + h_{69} & 0 & 0 & h_{66} & 0 \\
  h_{71} & 0 & h_{77} & 0 & h_{78} \\
  h_{85} + h_{89} & 0 & 0 & 0 & h_{88} \\
  h_{99} & h_{910} & 0 & h_{96} & h_{98} \\
  h_{109} & h_{1010} & 0 & h_{104} & 0 & 0
\end{bmatrix}
\]

for each two-colored star

for each spoke-hub pair \((h_s, h_u)\)

\[
H[s, u] \leftarrow B[s, color[h_u]]
\]

Substitution (acyclic coloring)

\[
\begin{bmatrix}
  h_{11} & h_{12} + h_{17} & 0 \\
  h_{21} + h_{23} + h_{25} & h_{22} & 0 \\
  h_{31} & h_{32} + h_{34} & h_{36} \\
  h_{41} + h_{410} & h_{44} & 0 \\
  h_{55} & h_{52} & h_{56} + h_{58} \\
  h_{65} + h_{69} & h_{69} & h_{66} \\
  h_{71} & h_{77} & h_{78} \\
  h_{85} & h_{87} + h_{89} & h_{88} \\
  h_{910} & h_{99} & h_{96} + h_{98} \\
  h_{1010} & h_{104} & h_{109}
\end{bmatrix}
\]

for each two-colored tree \(T\)

while \(T\) is non-empty

evaluate and delete “leaf” edges

Further reading:

Gebremedhin, Pothen, Tarafdar and Walther, Efficient computation of sparse Hessians using coloring and AD. INFORMS JOC, in press.
Outline

1. Models
2. Sequential algorithms
   - ColPack
3. A case study
4. Parallel algorithms
<table>
<thead>
<tr>
<th>General graph, $G = (V, E)$</th>
<th>Bipartite graph, One sided coloring, $G_b = (V_1, V_2, E)$</th>
<th>Bipartite graph, Bicoloring, $G_b = (V_1, V_2, E)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Distance-1 coloring $O(</td>
<td>V</td>
<td>d_1) = O(</td>
</tr>
<tr>
<td>• Distance-2 coloring $O(</td>
<td>V</td>
<td>d_2)$</td>
</tr>
<tr>
<td>• Star coloring* $O(</td>
<td>V</td>
<td>d_2)$</td>
</tr>
<tr>
<td>• Acyclic coloring $O(</td>
<td>V</td>
<td>d_2 \alpha)$</td>
</tr>
<tr>
<td>• Restricted star coloring $O(</td>
<td>V</td>
<td>d_2)$</td>
</tr>
<tr>
<td>• Triangular coloring* $O(</td>
<td>V</td>
<td>d_2)$</td>
</tr>
</tbody>
</table>

* more than one algorithm available; complexity of fastest algorithm shown
### Ordering techniques

<table>
<thead>
<tr>
<th>Ordering</th>
<th>Description</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Largest First</td>
<td>(v_i) has largest degree in sequence (v_i, v_{i+1}, \ldots, v_n)</td>
<td>sorted in non-increasing order of degrees in input graph (G)</td>
</tr>
<tr>
<td>Incidence Degree</td>
<td>(v_i) has largest back degree in sequence (v_i, v_{i+1}, \ldots, v_n)</td>
<td></td>
</tr>
</tbody>
</table>
| Smallest Last          | \(v_i\) has smallest back degree in sequence \(v_i, v_2, \ldots, v_i\) | • minimizes \(B\) over all orderings  
• \(B_{SL} + 1 = col(G)\) |
| Dynamic Largest First  | \(v_i\) has largest forward degree in sequence \(v_i, v_{i+1}, \ldots, v_n\) | \(B\) = max back degree over entire seq.  
\(B+1\) colors suffice to color \(G\). |

\(O(|E|)\)-time implementations possible for all four

*back degree*  \(\Rightarrow\)  *forward degree*  \(\Rightarrow\)  *degree*  \(\Rightarrow\)  *degree*
### ColPack: ordering capabilities

<table>
<thead>
<tr>
<th>General graph</th>
<th>Bipartite graph, One sided coloring</th>
<th>Bipartite graph, Bicoloring</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Natural</td>
<td>• Column Natural</td>
<td>• Natural</td>
</tr>
<tr>
<td>• Random</td>
<td>• Column Random</td>
<td>• Random</td>
</tr>
<tr>
<td>• Largest First</td>
<td>• Column LF</td>
<td>• LF</td>
</tr>
<tr>
<td>• Smallest Last</td>
<td>• Column SL</td>
<td>• SL</td>
</tr>
<tr>
<td>• Incidence Degree</td>
<td>• Column ID</td>
<td>• ID</td>
</tr>
<tr>
<td>• Dynamic LF</td>
<td>• Row Natural</td>
<td>• Dynamic LF</td>
</tr>
<tr>
<td>• Distance-2 LF</td>
<td>• Row Random</td>
<td>• Selective LF</td>
</tr>
<tr>
<td>• Distance-2 SL</td>
<td>• Row LF</td>
<td>• Selective SL</td>
</tr>
<tr>
<td>• Distance-2 ID</td>
<td>• Row SL</td>
<td>• Selective ID</td>
</tr>
<tr>
<td></td>
<td>• Row ID</td>
<td></td>
</tr>
</tbody>
</table>
ColPack: other functionalities

- Recovery routines
  - Jacobians
    - Unidirectional, Direct (via distance-2 coloring)
      - columnwise and rowwise
    - Bidirectional, Direct (via star bicoloring)
  - Hessians
    - Direct (via star coloring)
    - Substitution (via acyclic coloring)

- Graph construction routines
  - Various file formats supported
ColPack: organization

![Diagram of ColPack organization](image)

Further information & software download:
www.cscapes.org/coloringpage
A case study: Jacobian computation in Simulated Moving Beds

- SMBs useful in chromatographic separation
- Tested efficacy of 4-step procedure:
  - Used ADOL-C for steps S1 and S3, and ColPack for steps S2 and S4
  - Observed results for each step matched analytical results
  - Techniques enabled huge savings in runtime
    \[ \text{Time(Jacobian eval)} \approx 100 \times \text{Time(function eval)} \]

Further reading:
Gebremedhin, Pothen and Walther,
Exploiting sparsity in Jacobian computation: a case study in an SMB process.
Outline

1. Models
2. Sequential algorithms
   ◦ ColPack
3. A case study
4. Parallel algorithms
Parallelizing greedy coloring

- Desired task: parallelize Greedy such that
  - Speedup is proportional to number of processors
  - Number of colors used is roughly same as in serial
- Difficult task since Greedy is inherently sequential
- For distance-1 coloring, there exist several approaches based on Luby’s parallel algorithm for maximal independent set. Some drawbacks:
  - No actual parallel implementation
  - Many more colors than a serial implementation
  - Poor parallel speedup on unstructured graphs
- No practical parallel algorithms existed for distance-2 coloring, star coloring, etc
A generic parallelization technique

- **Partitioning:**
  - Break up the given problem into $p$ independent subproblems of almost equal sizes
  - Solve the $p$ subproblems concurrently using $p$ processors

  Main work 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

  Technique feasible as long as the resolution in the fourth step involves only “local” adjustments
Enhanced RP applied to greedy coloring: Basic features of the framework

- Exploits features of initial 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
### Specializations of the framework

<table>
<thead>
<tr>
<th>Category</th>
<th>Strategies/Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Color selection strategies</strong></td>
<td>• First Fit</td>
</tr>
<tr>
<td></td>
<td>• Staggered First Fit</td>
</tr>
<tr>
<td><strong>Coloring order</strong></td>
<td>• Interior before boundary</td>
</tr>
<tr>
<td></td>
<td>• Interior after boundary</td>
</tr>
<tr>
<td></td>
<td>• Interior interleaved with boundary</td>
</tr>
<tr>
<td><strong>Local vertex ordering</strong></td>
<td>• Various degree-based techniques</td>
</tr>
<tr>
<td><strong>Supersteps</strong></td>
<td>• Synchronous</td>
</tr>
<tr>
<td></td>
<td>• Asynchronous</td>
</tr>
<tr>
<td><strong>Inter-processor communication</strong></td>
<td>• Customized</td>
</tr>
<tr>
<td></td>
<td>• Broadcast-based</td>
</tr>
</tbody>
</table>
“Parameter” configuration

- How should the various options in the framework be set?
- Answer requires consideration of a complex set of factors, including:
  - Size and density of input graph
  - Number of processors
  - Quality of initial partitioning
  - Characteristics of platform on which implementation is run
- Determination bound to rely on experimentation
Lessons learned from experiments

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

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

- Highly unstructured graphs (e.g. a random graph):
  1. $s$ in the order of 100
  2. items 2 to 4 same as above
  3. broadcast-based communication
Sample experimental results

FBAC on Itanium 2

strong scaling

weak scaling

Itanium 2

Pentium 4
Software and further information

- Used the framework to design parallel algorithms for
  - distance-1 coloring of general graphs (discussed here)
  - distance-2 and restricted star coloring of general graphs, and partial distance-2 coloring of bipartite graphs (not discussed here)

- MPI-based implementations of all made available via Zoltan
  - www.cs.sandia.gov/Zoltan

- Further reading:
  - Bozdag, Catalyurek, Gebremedhin, Manne, Boman and Ozgunner, Distributed-memory parallel coloring algorithms for Jacobian and Hessian computation. *Submitted to journal*. (distance-2 and RS colorings)
Summary

- **Current accomplishments:**
  - Developed a **unifying theory** for sparse derivative computation
  - Designed **novel sequential algorithms** for distance-k, star, acyclic, and other coloring problems
  - C++ implementations assembled in a package called **ColPack**
    - ColPack also includes various ordering, recovery, and graph construction routines
  - Interfaced ColPack with the AD tool **ADOL-C**
  - Developed **family of parallel algorithms** for distance-1, distance-2, and restricted star coloring
    - Algorithms scale well for a hundred processors
    - Implementations made available via Zoltan

- **Future plans include:**
  - Integrate coloring software with tools in **OpenAD**
  - Extend **scalability** to tera- and petascale computation
  - Seek **collaborations** to “plug in” coloring technologies to enable CSE

- **Further information:**
  - Visit **www.cscapes.org**
Further readings put together


