Minimum-Mapping based Connected Components Algorithm

Zhihui Du (Presenter)
Oliver Alvarado Rodriguez
Fuhuan Li
Mohammad Dindoost
David A. Bader

This research is supported by NSF grant CCF-2109988
Connected Components Problem

• Graph partition
  • Given undirected graph \( G = \langle V, E \rangle = G_1 + G_2 + \ldots + G_k \)
    • All vertices in \( G_i \) are connected with each other (or single vertex)
    • \( G_i \) And \( G_j \) no overlap

• Importance
  • Graph Structure
  • Algorithm
Existing Methods (Abstract)

- **Graph Traversal**
  - $O(n)$ iterations
  - Label Propagation/BFS operations

- **Tree**
  - $n \cdot \log(n)$ iteration
  - Hooking-Compressing operation

- **Disjoint Set**
  - Approximate linear time (sequential)
  - Union-Find Operations
Minimum-Mapping based Contour Algorithm

• Contour line
  • Mapping the vertices to different contour lines
  • Give edge <u,v>
    • **Search**: minimum label of their ancestors
    • **Remap**: Update labels of descendants
  • Converge in log(n) time

• Feature
  • Simple Operations
  • Easy to Parallel

• Algorithm
  • Initialize the label array
  • Repeat
    • Forall edge=<u,v>
      • Minimum-mapping based on edge <u,v>
  • Until converge
Example

Converge in $\log(n)$ iterations
Algorithm Implementations

• Arkouda/Chapel Implementation
  • Contour
    • Variants (search steps, update methods)
      • C-1/C-2/C-S/C-CAS/C-Syn
  • FastSV

• High-Level Graph Package Implementation
  • LAGraph (C)
    • Contour
    • FastSV

• Low-Level Graph Package
  • Graph Based Benchmark Suite – GBBS (C++)
    • Contour
    • Simplified SV

Algorithm 1: Voltage-Mapping based Contour Algorithm

```plaintext
Contour(G) 
/* G = <E,V> is the input graph with edge set E and vertex set V. m = |E| is the total number of edges and n = |V| is the total number of vertices. */ 
1 for all i in 0..n-1 do
2     L[i] = i 
3     Lu[i] = i 
4 end
/* Initialize the label array L, Lu */
5 while (There is any label change in L) do
6     for all (e = (w,v) ∈ E) do
7         VO2(Lu, L, w, v)
8     end
9     L = Lu
10 end
11 return L
```
Experimental Results (number of iterations)

Table 1. Dataset 1 - Real-World and Synthetic Graphs

<table>
<thead>
<tr>
<th>Graph Type</th>
<th>Graph ID</th>
<th>Graph Name</th>
<th>m</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real-World Graph</td>
<td>1</td>
<td>loc-brightkite_edges</td>
<td>214078</td>
<td>58228</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>soc-Epinions</td>
<td>405790</td>
<td>75879</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>amazon0601</td>
<td>2443408</td>
<td>403394</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>com-youtube,ungraph</td>
<td>2987624</td>
<td>1134890</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>soc-LiveJournal1</td>
<td>68993773</td>
<td>4847570</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>kmer_V1r</td>
<td>232750452</td>
<td>214005017</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>kmer_A2a</td>
<td>180292886</td>
<td>170728175</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>uk-2002</td>
<td>261787258</td>
<td>18484117</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>uk-2005</td>
<td>783027125</td>
<td>39454746</td>
</tr>
<tr>
<td>Synthetic Graph</td>
<td>10</td>
<td>rgg_n_2_21_s0</td>
<td>14487995</td>
<td>2097148</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>rgg_n_2_22_s0</td>
<td>30359198</td>
<td>4194301</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>rgg_n_2_24_s0</td>
<td>132557200</td>
<td>16777215</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>kron_g500-logn16</td>
<td>2456071</td>
<td>55321</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>kron_g500-logn18</td>
<td>10582868</td>
<td>210155</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>kron_g500-logn20</td>
<td>44619402</td>
<td>795241</td>
</tr>
</tbody>
</table>

C-2: search two steps
C-S: Simplified minimum-mapping
C-CAS: compare-and-swap operation for update
C-1: search one step
C-Syn: synchronization before updates
FastSV: state-of-the-art tree-based method
Experimental Results (performance)

Contour VS FastSV

<table>
<thead>
<tr>
<th>Graph ID</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.125</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>13</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>8</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
</tr>
</tbody>
</table>
Multi-Locale Results

**Iterations of Different Methods**

Number of Iterations

Graph ID

- Contour
- C-1
- C-2
- C-S
- C-CAS
- C-Syn
- FastSV

**Speedup**

Graph ID

- Contour
- C-1
- C-2
- C-S
- C-CAS
- C-Syn

NJIT New Jersey Institute of Technology
Other Implementations (delaunay_n20)

- Chapel
  - FastSV: 0.290952s
  - Contour: 0.038154s

- LAGraph implementation (C+GraphBLAS)
  - FastSV needs: 0.0226186s
  - Contour just like FastSV

- Graph Based Benchmark Suite (GBBS)
  - Optimized SV: 0.022097s
  - Contour: 0.012859s
Conclusion

• Contour algorithm
  • simple, easy to parallelize and high-performance for connected components
  • Converge in O(log(n)) iterations

• How Chapel can affect the performance
  • Compared with High-Level LAGraph(GraphBLAS) package (C) (Vector, Matrix)
    • LAGraph/GraphBLAS cannot exploit fine and flexible parallelism like Chapel
    • Chapel has a performance overhead
  • Compared with the Graph-Based Benchmark Suite (GBBS) package (C++)
    • GBBS cannot support distributed parallelism like Chapel
    • Chapel’s overhead is relatively high
Acknowledgement

We appreciate the help from the Chapel and Arkouda community when we integrated the algorithms into Arkouda. This research was funded in part by NSF grant number CCF-2109988.
Thank You!

Q&A