Mix-and-Match: A Model-driven Runtime Optimisation Strategy for BFS on GPUs

Merijn Verstraaten\textsuperscript{1,2}, Ana Lucia Varbanescu\textsuperscript{1} & Cees de Laat\textsuperscript{1}

\textsuperscript{1} University of Amsterdam

\textsuperscript{2} Netherlands eScience Center

November 12, 2018
Breadth-First Search: Implementations

**Edge-centric**

**Vertex Push**

**Vertex Pull**

Useless Frontier
Thread

Useful Frontier
Thread

Updated Node

Frontier Node

Accessed Node

Mix-and-Match BFS
Relative Performance of Implementations

There is no “best”!

M. Verstraaten, A.L. Varbanescu & C. de Laat

Mix-and-Match BFS
Sticking to one implementation costs us!
Choosing the best algorithm:

- Depends on algorithm + platform + graph
- Predict “best” implementation each level

Challenge?

How to model the algorithm + graph + platform

Is it worth it? It depends on...

- …gain
- …prediction cost
- …data representation for implementations
Modelling the Problem

Analytical model:
1. Build a parametrised work model of the algorithm
2. Use graph properties as parameters
3. Calibrate using hardware microbenchmarking

Result: Prediction accuracy below 50%…

What’s going wrong?
Intuition & experience say this should work.

Results say it doesn’t! But why?

**Problem:** Sequential workload → Parallel GPU execution

**But:** Best implementation stable over several GPU generations

**What now?**
Training Parameters:
- Degree distribution
- Frontier size
- Percentage discovered
- Vertex count
- Edge count

Pros:
- Black-box approach
- Fast! (Training & prediction)
- Variable importance!

Cons:
- Can overfit on non-uniform parameters
- Bad with large numbers of parameters
Two Questions

Do the models actually work?

Do the models match our intuitions?
Trained Models

Feasibility:
Average Prediction Time: 144 ns (σ = 165 ns)
Minimum BFS Step: 20 ms
(Re)loading graph representation: Stupidly slow

Classic time-space trade-off.
### Comparison with State-of-the-Art: Best & Worst

<table>
<thead>
<tr>
<th>Graph</th>
<th>Mix-and-Match</th>
<th>Gunrock</th>
<th>Lonestar</th>
<th>Non-switching Best</th>
<th>Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Even better if we include Gunrock in model?

M. Verstraaten, A.L. Varbanescu & C. de Laat

Mix-and-Match BFS
Overall Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>1–2×</th>
<th>&gt;5×</th>
<th>&gt;20×</th>
<th>Average</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mix-and-Match</td>
<td>92%</td>
<td>2.5%</td>
<td>0.4%</td>
<td>2.04×</td>
<td>498×</td>
</tr>
<tr>
<td>Non-switching Best</td>
<td>65%</td>
<td>8%</td>
<td>0%</td>
<td>2.44×</td>
<td>37×</td>
</tr>
<tr>
<td>Edge List</td>
<td>49%</td>
<td>22%</td>
<td>2.2%</td>
<td>4.16×</td>
<td>61×</td>
</tr>
<tr>
<td>Rev. Edge List</td>
<td>39%</td>
<td>33%</td>
<td>8.8%</td>
<td>7.04×</td>
<td>108×</td>
</tr>
<tr>
<td>Vertex Pull</td>
<td>16%</td>
<td>58%</td>
<td>30%</td>
<td>48.41×</td>
<td>2,495×</td>
</tr>
<tr>
<td>Vertex Push</td>
<td>23%</td>
<td>53%</td>
<td>28%</td>
<td>55.61×</td>
<td>1,980×</td>
</tr>
<tr>
<td>Vertex Push Warp</td>
<td>18%</td>
<td>25%</td>
<td>4.9%</td>
<td>5.42×</td>
<td>88×</td>
</tr>
</tbody>
</table>

Averaged over 248 KONECT graphs.
Parameter importance matches intuition

Investigating “poor” predictions reveals new insights

Not investigated (yet):

- Handle implementations with similar results
- Minimising training data
- Model portability across datasets, hardware & algorithm
- Relating BDTs to analytical model
Summary

The Good:

- Prediction works!
- Predictions are fast enough at runtime
- Our Mix-and-Match outperforms state-of-the-art (on average)
- Models provide new insights

The Bad:

- Training set too big
- Training set non-uniformity
Takeaway

No single best implementation for irregular GPU algorithms

Large potential performance gains for graph algorithms

Mix-and-Match is a generalisation of direction-optimised BFS

Significant performance improvement for many graphs

Method applicable to any BSP graph algorithm

Will revisit analytical models using decision tree models

Questions?