Exploring Graph Colouring Heuristics in GraphLab

Open Source Project

Philip Leonard

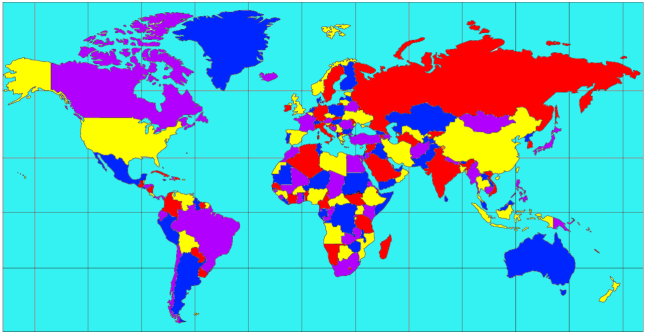
December 1st, 2014

Significance

Applications;

- Map colouring (four colouring problem)
- The timetabling problem (various scheduling problems)
- GSM Frequency assignment

NP-complete: reducible to lots of other problems, like graph covering.



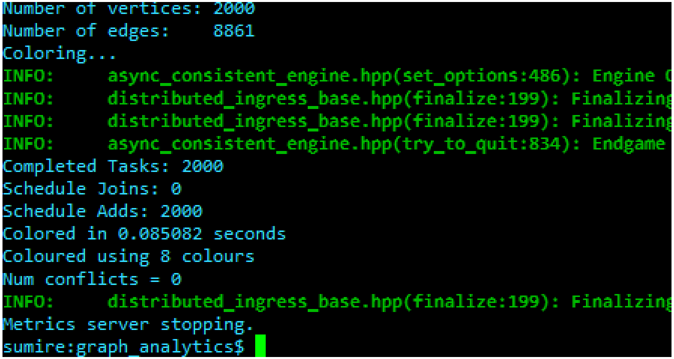
Similar Work

Graph Analytics Toolkit

GraphLab includes a greedy “simple colouring” heuristic [2];

- Employs first fit selection
- Vertex coloured with smallest non conflicting colour
- No decision process behind vertex selection

Existing GraphLab Toolkit



```
Number of vertices: 2000
Number of edges:    8861
Coloring...
INFO:     async_consistent_engine.hpp(set_options:486): Engine C
INFO:     distributed_ingress_base.hpp(finalize:199): Finalizing
INFO:     distributed_ingress_base.hpp(finalize:199): Finalizing
INFO:     async_consistent_engine.hpp(try_to_quit:834): Endgame
Completed Tasks: 2000
Schedule Joins: 0
Schedule Adds: 2000
Colored in 0.085082 seconds
Coloured using 8 colours
Num conflicts = 0
INFO:     distributed_ingress_base.hpp(finalize:199): Finalizing
Metrics server stopping.
sumire:graph_analytics$
```

How can we pick the next vertex more effectively?

Exploring Other Heuristics

Possible vertex selection heuristics proposed in [1];

- Highest degree vertex first $O(n^2)$
- Incidence degree ordering $O(n^2)$, picks the vertex with the largest coloured neighbourhood first.
- Saturation degree ordering $O(n^3)$, picks the vertex with the most differently coloured neighbourhood first.

[1] combines highest degree and saturated degree ordering approaches.

Graphlab allows for asynchronous **dynamic scheduling**

What Will Be Produced?

- Extended GraphLab toolkit
- A number of greedy heuristic methods
- A tradeoff version might be possible
  - i.e. use degree based scheduling for first $\frac{|V|}{x}$ colourings, then resort back to random selection.
- An extensive comparison of these methods against the existing toolkit
- Comparison will look at natural vs random and runtime vs chromatic number trade-offs

Why?

- The existing toolkit picks performance over optimality.
- Currently users can't experiment with tradeoffs.
- This extended toolkit will allow users to leverage computation for more optimal colourings

Plan

- Implement heuristic methods proposed in [1].
- Combine and alter these methods in order to find the optimal approach
- Conduct a comparison
- Given time, explore further heuristic methods and repeat cycle.
- Write Report

References

Hussein Al-Omari and Khair Eddin Sabri
New Graph Coloring Algorithms
American Journal of Mathematics and Statistics 2 (4): 739-741, 2006.

GraphLab
Graph Analytics Simple Colouring Toolkit
http://docs.graphlab.org/graph_analytics.html