Performance Studies of the Blossom V Algorithm

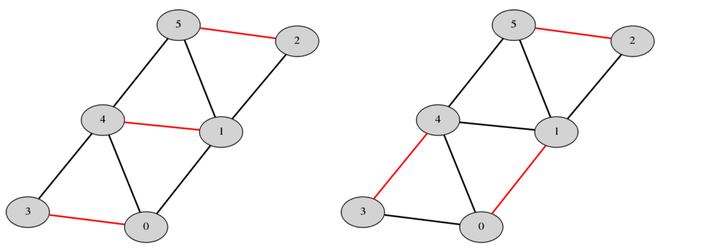
UMBC REU Site: Interdisciplinary Program in High Performance Computing
Changling Huang[1], Christopher C. Lowman[2], Brandon E. Osborne[3], Gabrielle M. Salib[4],
Graduate assistants: Ari Rapkin Blenkhorn[4], Jonathan S. Graf[5], Samuel Khuvis[5],
Faculty mentor: Matthias K. Gobbert[5], Clients: Tyler Simon[6] and David Mountain[6]
[1]Rutgers U., [2]UMCP, [3]Austin Peay State U., [4]CSEE, UMBC, [5]Math & Stat, UMBC, [6]Lab. for Phys. Sci.

Objective

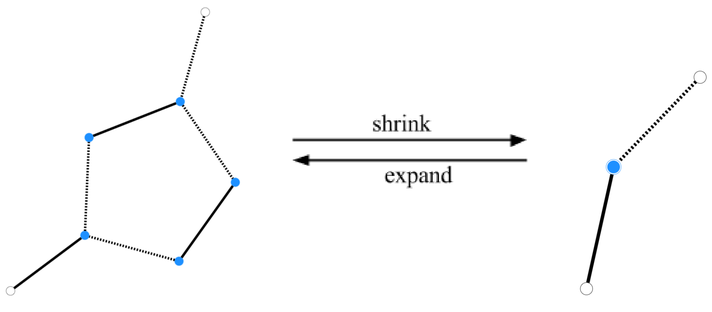
To better understand the performance capabilities of the Blossom V algorithm and highlight potential approaches for improvement, we conducted performance studies for a variety of graphs. We varied the number of nodes, graph density, and the range of edge weights and recorded the initialization time, total execution time, and total memory usage for each graph.

Blossom V Algorithm

The Blossom algorithm was first introduced in 1965 and has been incrementally improved over time. The most recent version, Blossom V, computes a perfect matching of minimum cost of a graph. Below are examples of perfect matchings, with edges in the matching highlighted in red.



A critical component of the algorithm involves the use of blossoms. A *blossom* is a cycle of the graph consisting of $2k+1$ edges, exactly k of which belong to a matching. Blossoms are *shrunk* and *expanded* in order to perform efficient searches on a reduced graph. These operations are represented below.



Timing Results

We recorded wall times for graphs with various numbers of nodes n and graph densities d. In the tables below, Init. represents initialization time, which includes updating optimization variables and assigning matchings. Total represents total execution time of the algorithm.

The results below illustrate the effects of increasing graph density and increasing the range of the edge weights.

Time in seconds for $n = 2^{15}$ nodes with weights 1 to 100

d	Init. (s)	Total (s)
0.125	6.074	8.262
0.250	11.207	13.006
0.500	27.505	29.046
1.000	47.547	49.087

Time in seconds for $n = 2^{15}$ nodes with density $d = 1$

Weight range	Init. (s)	Total (s)
1 to 10^2	47.547	49.087
1 to 10^4	729.372	878.405
1 to 10^6	2649.447	2670.155
1 to 10^8	2677.173	2700.346

We observed that the times varied for different ranges of weights and investigated whether this was a matter of magnitude. We scaled down the same graphs by their maximum weights to obtain smaller weights.

Scaling speedup (unscaled/scaled time) for 2^{15} nodes with density $d = 1$

Weights	Speedup (I)	Speedup (T)
10^{-2} to 1	0.786	0.792
10^{-4} to 1	0.858	0.914
10^{-6} to 1	0.920	0.924
10^{-8} to 1	0.943	0.945

Memory Results

For each graph where initialization time and total execution time were recorded, we also recorded total memory usage.

Memory used for $n = 2^{15}$ nodes

d	Memory (GB)
0.125	7.251
0.250	14.498
0.500	28.991
1.000	57.981

There were no major disparities between repeated trials of graphs with the same number of nodes and edges, regardless of the range of edge weights.

Conclusions

As graph density increases, initialization time and total execution time increase.

As the range of edge weights increases, initialization time and total execution time increase.

Scaling down the edge weights has no significant effect on initializtion time and total execution time.

Memory usage largely depends on the number of nodes and edges in a graph.

References

- Kolmogorov, *Math. Prog. Comp.*, 2009
- Full technical report: HPCF–2015–26
  `hpcf.umbc.edu` > Publications

Acknowledgments

REU Site: `hpcreu.umbc.edu`
NSF, NSA, DOD, UMBC, HPCF, CIRC