Graph Coloring Using State Space Search

Sandeep Dasgupta* Tanmay Gangwani† Mengjia Yan‡

October 1, 2014

1 Definitions

The graph coloring problem deals with assigning each vertex of the graph with a color such that the adjacent vertices have distinct colors. Many applications have been shown to be reducible to the graph coloring problem. Some of these include 1) register allocation in the back-end of a complier, and 2) scheduling of multiple tasks to resources under various constraints. Graph coloring is computationally hard. It is NP-complete to determine if a given graph admits k-coloring (except for k=1 or k=2).

State space search is a technique where successive configurations (a.k.a states) are explored until the state with desired property is found. The different states form a tree where child nodes are produced from the parent node by changing some part of the parent state. In typical problems, exploring the entire state space graph is impractical due to execution time and memory constraints. Application of heuristics has been shown to be effective in pruning the search space.

2 Parallel Graph Coloring

Large-scale systems with distributed memory are a natural fit to solving the computationally complex graph coloring problem. There are two main parallelization strategies, namely domain decomposition and state space exploration. In domain decomposition, the graph is divided into sub-graphs, and each of these is assigned to a processor. The sub-graph is colored locally at each processor. The algorithm operates in time-steps, where at the end of each time-step, the processors exchange coloring information with each other to resolve the conflicts for the vertices on the sub-graph boundaries. Such an approach is outlined in [1]. State space exploration takes a different approach to dividing work between processors. A *state* in the state space tree is a partially colored input graph, and child states are produced from parent states by coloring one of the uncolored vertices. A solution is found if one of the leaves if a legitimately colored input graph. Each processor is responsible for exploration of a part of the state space tree. This idea has been used in [4, 3, 2].

3 Project Goals

We plan to leverage the state space search model for implementing graph coloring in parallel in Charm++. Some of the challenges for efficient exploration of space by *chares* include intelligent pruning of the state space, load balancing, grain-size control and low-overhead communication

*Electronic address: sdasgup3@illinois.edu
†Electronic address: gangwan2@illinois.edu
‡Electronic address: myan8@illinois.edu

between chares. We plan to evaluate multiple options for each of these, and hopefully come up with design decisions which would work for a large category of real-life graph applications.

4 References

References

[1] E. Boman, D. Bozda, U. Catalyurek, A. Gebremedhin, and F. Manne, *A scalable parallel graph coloring algorithm for distributed memory computers*, in Euro-Par 2005 Parallel Processing, J. Cunha and P. Medeiros, eds., vol. 3648 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2005, pp. 241–251.

[2] L. V. Kalé, B. H. Richards, and T. D. Allen, *Efficient parallel graph coloring with prioritization*, in Proceedings of the International Workshop on Parallel Symbolic Languages and Systems, PSLS '95, London, UK, UK, 1996, Springer-Verlag, pp. 190–208.

[3] V. Saletore and L. Kale, *Consistent linear speedups for a first solution in parallel state-space search*, in Proceedings of the AAAI, August 1990, pp. 227–233.

[4] Y. Sun, G. Zheng, P. Jetley, and L. V. Kalé, *ParSSSE: An Adaptive Parallel State Space Search Engine*, Parallel Processing Letters, 21 (2011), pp. 319–338.