Deterministic Parallel Graph Coloring with Symmetry Breaking

Per Normann, Johan Öfverstedt

October 2015

Abstract In this paper we propose a deterministic parallel graph coloring algorithm that enables Multi-Coloring in parallel for sparse undirected graphs by coarse-grained segmentation and symmetry breaking. The proposed algorithm is implemented and tested on standard problem instances from engineering applications and benchmarked against various deterministic graph coloring algorithms. Quantified results show that the proposed algorithm is competitive or better than the sequential Multi-Coloring algorithm with respect to execution time on multi-core architectures. The upper bound on the number of colors is guaranteed to be the same as for the Multi-Coloring algorithm.

1 Introduction

In computing and numerical science the vertices and edges of a graph are commonly denoted as V and E, the degree of a graph as Δ. Graph coloring is the problem of associating every vertex with a color such that each pair of adjacent vertices are assigned distinct colors. Colors are generally denoted by integers. In Figure 1.1 the vertices have been assigned integers according to the constraints of graph coloring.

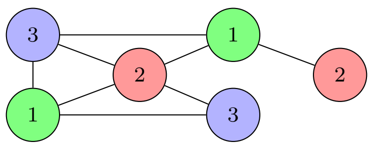

Figure 1.1: A graph with vertices colored such that no adjacent vertices share the same color. Colors are generally denoted by integers.

Finding an optimal coloring that minimizes the number of colors is a NP-hard problem and therefore intractable in general, requiring the use of heuristics [1]. Contemporary applications employing deterministic graph coloring predominantly implements some variation of the Multi-Coloring algorithm [2]. The Multi-Coloring algorithm constitutes a formidable benchmark for graph coloring both in execution time and the quality of the coloring [3].

The Multi-Coloring algorithm is sequential and does not allow for efficient use of contemporary multi-core hardware. We are aware of two different techniques of deterministic parallel graph coloring suited for contemporary machines. First there are parallelizations of Luby-Jones methods [4] where independent sets of vertices are formed locally in the graph to be safely colored in parallel and further a novel technique was proposed by us in [5]. In this paper we propose a new deterministic parallel graph coloring algorithm for sparse undirected graphs.

2 New Graph Coloring Algorithm

The proposed algorithm is a deterministic parallel graph coloring algorithm that enables Multi-Coloring in parallel for sparse undirected graphs by coarse-grained segmentation and symmetry breaking. Shared memory architecture and a data structure supporting segmentation without an initial traversal are assumed. The general idea is to split the graph into a set of segments S, each of which are touched by at most one processor in a given coloring round. Let $v_{i,j}$ denote an adjacent vertex of a vertex v_i in a graph. Let S_k and S_m be segments of the graph. All v_i with an adjacent vertex $v_{i,j}$ satisfying the predicate in Equation 2.1 are unsafe to color in parallel.

$$\texttt{Unsafe}(v_i) \coloneqq \begin{cases} \exists S_k | v_i \in S_k \\ \exists S_m | v_{i,j} \in S_m \\ m < k \end{cases} \tag{2.1}$$

The vertices of a segment are traversed and safe vertices are colored while the coloring of unsafe vertices is deferred to a future coloring round. If a vertex is colored it is excluded from its segment at the end of the coloring round. This procedure is repeated until all vertices are colored.

In Figure 2.1 safe vertices of a graph are colored in parallel according to the rules of symmetry breaking in Equation 2.1.

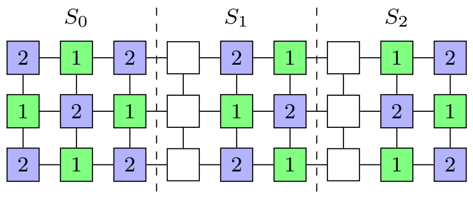

Figure 2.1: The example graph is segmented into S_0, S_1 and S_2. All vertices that do not satisfy the symmetry breaking predicate in Equation 2.1 are colored.

The proposed algorithm is deterministic in the sense that the coloring is always the same for a given graph and initial segmentation. The upper bound on the number of colors is inherited from the Multi-Coloring algorithm, i.e. $\Delta + 1$. This is guaranteed because the proposed algorithm is functionally equivalent of coloring the graph sequentially in a sorted ordering based on the index of the outer iteration when a given vertex is first deemed safe to color according to the rules of symmetry breaking as stated in Equation 2.1.

In each coloring round the non-empty segment with the lowest index will have no vertex satisfying the predicate in Equation 2.1, hence it will be colored completely. In the worst case all segments are colored and the algorithm terminates after $|S|$ rounds. The complexity of each coloring round is $\mathrm{O}\big(|V|+|E|\big)$ and the complexity of the synchronization round is $\mathrm{O}\big(|S|\big)$.

The proposed algorithm is presented in pseudo code in Algorithm 1.

Algorithm 1 Deterministic Symmetry Breaking Algorithm

```
 1: while $\exists c_i = 0$ do
 2:     for $\forall v_i$ do                              ▷ In parallel over segments $S$.
 3:         if $c_i = 0$ then
 4:             $C \leftarrow \varnothing$                 ▷ Reset color set $C$.
 5:             for $\forall v_{i,j}$ adjacent to $v_i$ do
 6:                 if $u_{i,j}$ then                     ▷ See constraints (2.1).
 7:                     $v_i$ is unsafe                   ▷ Do not color $v_i$
 8:                     break
 9:                 else
10:                     $C \leftarrow c_j \cup C$         ▷ Form set of forbidden colors $C$.
11:                 end if
12:             end for
13:             if $v_i$ not unsafe then
14:                 $c_i \leftarrow \min(c \notin C)$     ▷ Color $v_i$ with first color not in $C$.
15:             end if
16:         end if
17:     end for
18:     Synchronization round.
19:     Remove $v_i$ from corresponding segment if $c_i \neq 0$.
20: end while
```

3 Implementation and Test Configuration

For testing and benchmarking the algorithms in Table 3.1 are implemented in plain C/C++ using the low-level pthreads-library for multi-threading within a single testing framework. The framework and implementations are crafted for efficiency and performance to reduce noise and systematic error from the measured run times and resulting speed-up calculations.

Table 3.1: Implemented algorithms.

Sequential greedy Multi-Coloring
Deterministic hashing-based Luby-Jones
Proposed symmetry breaking algorithm

The test matrices used in this work are found on the matrix server[1] of the Department of Computer and Information and Engineering at the University of Florida. A summary of the features of the test matrices is presented in Table 3.2.

Table 3.2: Specifications of test graphs. The number of edges E and vertices V are scaled 10^{-6}, Δ is the degree of the graph.

	Graph specifications.			
Matrix	V	E	Δ	$avg\Delta$
apache2	0.7	4.8	7	5.7
bone010	1.0	71.7	80	71.6
ecology2	1.0	5.0	4	4.0
Flan_1565	15.6	117.4	80	74.0
G3_circuit	1.6	7.7	5	3.8
Geo_1438	1.4	63.2	56	41.9
Hook_1498	1.5	60.9	92	39.7
ldoor	1.0	46.5	76	46.9
msdoor	0.4	20.2	76	46.7
offshore	0.3	4.2	30	15.3
parabolic_fem	0.5	3.7	6	6.0
pwtk	0.2	11.6	179	52.4
Serena	1.4	64.5	248	45.4
StocF-1465	1.5	21.0	188	13.3
thermal2	1.2	8.6	10	6.0

All test matrices in Table 3.2 are colored using the algorithms in Table 3.1 and the results are compared with respect to execution time and color quality.

In [6] a hashing-based Luby-Jones scheme is presented as the currently best performing Luby-Jones scheme, hence we have selected this scheme as a representative of the Luby-Jones family of graph coloring algorithms. The sequential greedy Multi-Coloring algorithm is chosen based on the assumption that it is currently the best performing deterministic graph coloring algorithm [3].

The test graphs are converted to the CSR-format, which is described in [2], prior to the coloring procedure. In our implementation this ensures good access and locality of rows in test matrices, i.e. the sets of neighbouring vertices of each vertex. This choice of graph data format is also backed up by its ubiquity in numerical applications, its compact representation and its good memory locality [2].

The distributed segments in the proposed algorithm are represented by bounded intervals. At the end of each coloring round the considered segments are shrunk to minimum sub intervals containing all uncolored vertices. This constitutes a trade-off between bookkeeping overhead and inter processor visibility of progress.

The heuristic selected to color internal vertices in this work is the greedy Multi-Coloring algorithm. One of the major implementation decisions for this heuristic is the data structure used to maintain the set of neighbouring colors while searching a vertex's neighbourhood to compute its color. We implement a `BITSET` structure for this, thereby taking advantage of the target hardware platform's ability to efficiently store

[1]http://www.cise.ufl.edu/research/sparse/MM/ - Accessed: 2015-06-08

and operate on 64 colors as a single 64-bit integer due to it being a 64-bit architecture. This yields worst case time complexity of $O(1)$ for the `Insert`-operation and $O(\Delta)$ for the `FindFirst`-operation. `FindFirst`, although linear in complexity, has a small constant given the previously discussed implementation which is important for the practical purpose of being used in this algorithm. The internal bookkeeping memory required to operate this data structure is $O(\Delta)$.

The hardware on which the tests are conducted is Tussilago at the Division of Scientific Computing at Uppsala University, CPU: AMD Opteron (Bulldozer) 6282, 2.6 GHz, 16-cores, dual socket.

The selected test problems are large enough to heavily reduce significance of timer precision noise in the measured execution times. Time measurements are implemented by enclosing the algorithm with calls to `gettimeofday`.

Empirical correctness testing consists of a conflict and invalid color detection procedure which is executed after every run. Determinism and consistency between runs are verified by generating a hash-code from the resulting color sequence.

4 Results

A new high performance deterministic parallel graph coloring algorithm is proposed in this paper. The proposed algorithm enables Multi-Coloring in parallel for sparse undirected graph by coarse-grained segmentation and symmetry breaking. Table 4.1 summarizes the measured benchmark results.

Table 4.1: Performance of the proposed algorithm compared with implementations of Multi-Coloring (MC) and Luby-Jones (LJ). Time is in milliseconds.

	MC		LJ		Proposed algorithm	
Matrix	*Time*	*Colors*	*Time**	*Colors*	*Time**	*Colors**
apache2	17	3	207/103/49	18	28/14/8	3/4/4
bone010	160	39	10132/5421/2784	159	222/115/66	39/45/48
ecology2	21	2	174/87/41	12	35/16/9	2/4/4
Flan_1565	260	42	15585/8095/4224	149	398/209/111	42/42/45
G3_circuit	36	4	304/165/81	15	55/36/22	4/4/5/5
Geo_1438	161	29	7056/3714/1948	90	188/103/56	29/33/33
Hook 1498	156	30	6762/3492/1880	95	182/98/53	30/33/33
ldoor	115	42	5019/2657/1444	96	138/139/127	42/42/42
msdoor	50	42	2176/1147/610	95	59/53/58	42/42/42
offshore	16	12	289/148/76	39	18/15/15	12/13/13
parabolic_fem	17	5	163/103/57	17	21/29/29	5/5/6
pwtk	28	48	1378/713/291	128	32/20/13	48/48/48
Serena	160	36	7419/3973/2075	101	190/119/71	36/39/36
StocF-1465	80	11	1296/669/350	36	93/52/29	11/11/11
thermal2	53	7	420/218/116	18	61/41/29	7/7/7

* Figures presented as: 1/2/4-threads.

The overall execution time of the proposed algorithm is never hurt significantly by the parallelization, but some matrices do not exhibit speed-up. As expected the color quality achieved by the proposed algorithm satisfies the $\Delta + 1$ bound.

Figure 4.1 displays the speed-up achieved by an implementation of the proposed algorithm on the set of test graphs used in this work. The test graphs in Figure 4.1a show good speed-up and the test graphs in Figure 4.1b show minor speed-up.

The parallel overhead, calculated as the execution time of the proposed algorithm on one thread divided by the execution time of the implemented Multi-Coloring algorithm, measures 12-67% for the test matrices.

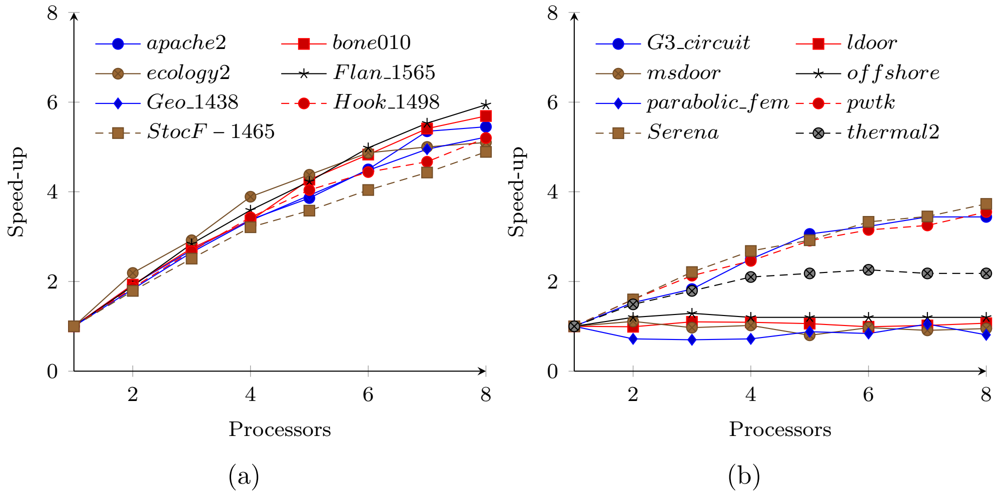

Figure 4.1: Parallel speed-up achieved by the proposed algorithm plotted with trend lines.

5 Discussion

In this work we show that efficient deterministic parallel graph coloring of sparse matrices on shared memory architectures is possible and an algorithm is proposed. It should be noted that there are local graph structures that are problematic for the proposed algorithm in the sense that they prevent parallel progress. One such problematic graph structure is exemplified in Figure 5.1. The vertices are in a path through multiple segments, hence preventing parallel coloring.

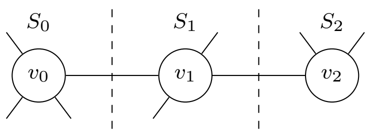

Figure 5.1: The vertices v_0, v_1 and v_2 are placed in different segments S_0, S_1 and S_2. The structure displayed requires three outer iterations as v_1 is dependant on v_0. v_2 is in turn dependant on v_1 and indirectly v_0.

A future field of investigation is the possibility of combining symmetry breaking with other parallel coloring schemes such as [5] and to explore the possibility to implement the proposed algorithm for distributed memory architecture. In addition to this, the effects of fine tuning the granularity of the segmentation technique in the proposed algorithm is not investigated in this work.

Bibliography

[1] Garey, M.R. Johnson, D.S., "Computers and intractability," *Freeman W.H. New York*, 1979.

[2] Saad, Y., "Iterative methods for sparse linear systems," *Society for Industrial and AppliedMathematics*, 2003.

[3] Allwright, R. Bordawekar, P.D. Coddington, K. Dincer, C. Martin, A., "A comparision of parallel graph coloring algorithms," *Cite Seer*, 1995.

[4] Luby, M., "A simple parallel algorithm for the maximal independent set problem," *SIAM Journal on Computing*, 1986.

[5] Normann, P. Öfverstedt, J., "Deterministic parallel graph coloring with hashing," *Technical report series at Division of Scientific Computing, Department of Information Technology, Uppsala University*, 2015.

[6] Cohen, J. Castonguay, P, "Efficient graph matching and coloring on the gpu," *Lectureslides*, 2012.