Parallel Graph Coloring

Andrea Cavallo, Gioele Scaletta

System and Device Programming
Politecnico di Torino

September 3rd, 2021

Table of Contents

Table of Contents

Introduction

- **Goal**: test sequential and parallel graph coloring algorithms, analyzing their performances in terms of time, number of colors used and memory occupation
- **Language**: C++
- **HW support**: i7-8550U, 1.80 GHz, 16 GB RAM
- **Operating System**: Windows
- **Reference**: Allwright et al., 1995

Table of Contents

Data Structure

```
vector<int> _colors, _weights, _tmp_degree, _new_colors, _new_weights;

/* one single array containing adjacencies for each node (one vector for each node) */
vector<vector<int>> _edges;
```

Figure: Data structure from class *graph*

- adjacency list with vectors (one vector with adjacencies for each node)
- additional vectors for colors, weights and other information necessary for the algorithms

Table of Contents

Algorithms - 1

Greedy

Algorithm 1 Greedy algorithm

Given $G = (V,E)$
$n = |V|$
choose a random permutation of the vertices V
for i=1 to n **do**
  select v_i
  C = colors of all colored neighbors of v_i
  c = smallest color not in C
  color v_i with color c
  $U = U - v_i$
end for

- sequential
- fast
- usually bad solutions

Jones-Plassman

Algorithm 2 Jones-Plassman

assign random weights to each node w
$U := V$
while $|U| > 0$ **do**
  for all vertices $v \in U$ **do** in parallel
    $I := \{v$ such that $w(v) > w(u)$ for all neighbors $u \in U\}$
    for all vertices $v' \in I$ **do** in parallel
      $S := \{$colors of all neighbors of $v'\}$
      $c(v') :=$ minimum color not in S
    end for
  end for
  $U = U - I$
end while

- parallel
- fast
- usually bad solutions

Algorithms - 2

Largest Degree First

- same as JP
- weights are degrees of nodes
- better solutions than JP

Smallest Degree Last

- more complex weight assignment
- better solutions than JP and LDF
- slower

Algorithm 3 Assignment of weights in SDL

$k = 1$
$i = 1$
$U := V$
while $|U| > 0$ **do**
  while $\exists$ vertices $v \in U$ with $d^U(v) \leq k$ **do**
    $S := \{$all vertices v with $d^U(v) \leq k\}$
    for all vertices $v \in S$ **do**
      $w(v) = i$
    end for
    $U := U - S$
    $i := i + 1$
  end while
  $k = k + 1$
end while

Table of Contents

Parallelization strategy

- Each thread is assigned a group of nodes
- Coloring is splitted in two distinct phases:
  - find nodes to color (check whether they are local maxima)
  - color the nodes (assign the minimum available color)
- Three different implementations:
  - **No threadpool**: one main thread creates worker threads that either find nodes to color or color them, and collects them after each iteration.
  - **Threadpool**: worker threads are created at the beginning of the algorithm. The main thread schedules jobs that are executed by workers. When the coloring is completed, the threads terminate.
  - **Find and color**: each job determines which nodes can be colored, stores them in a local queue and then colors them.

Table of Contents

Synchronization strategy

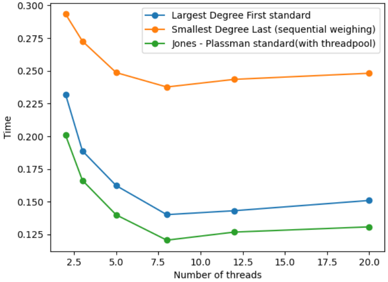
Three cases:

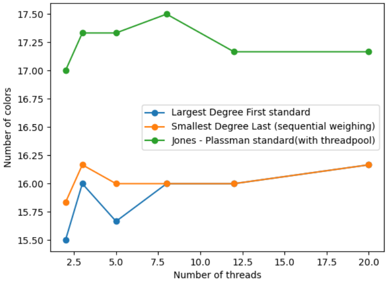
- **No threadpool**: condition variable with counter of active threads to keep the number of active threads constant
- **Threadpool**:
  - condition variable to manage the queue of jobs
  - condition variable with counter of scheduled jobs to wait for their termination
- **Find and color**: condition variable with counter of active jobs to synchronize threads after finding nodes to color (like a barrier). The same condition variable is used, with a different counter, by the main thread to wait for the termination of all jobs

Table of Contents

Politecnico di Torino

Number of threads



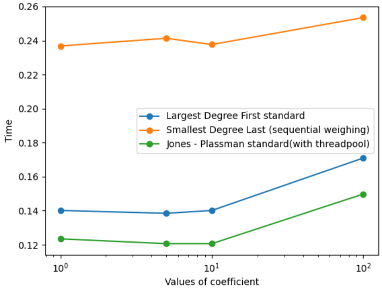


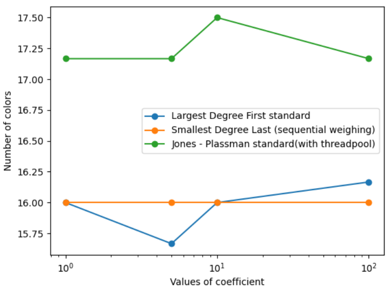
- best time for 8 threads
- better coloring for less than 8 threads
- final choice: 8 threads (equal to the hardware concurrency)

Hyperparameter optimization - 2

Number of nodes per thread

$$nodes\_per\_thread = \frac{nodes\_in\_the\_graph}{coef * number\_of\_threads}$$

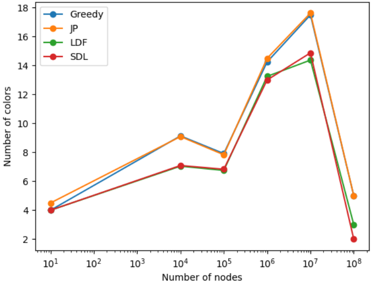


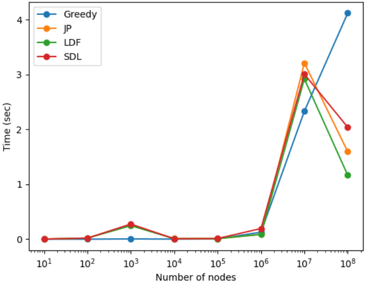


- not so relevant
- final choice: $coef = 10$

Table of Contents

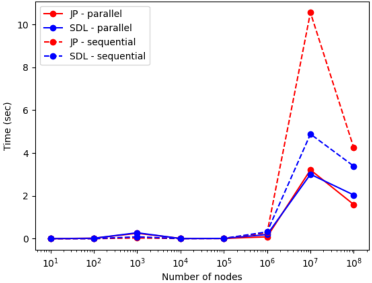
Results - 1

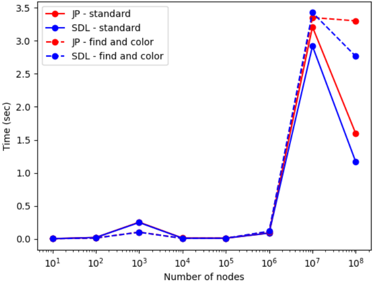




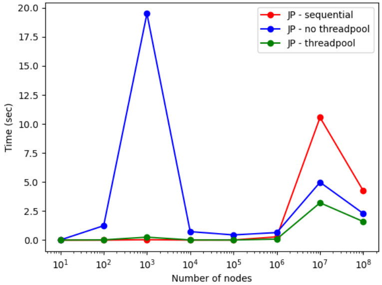
- LDF and SDL find better coloring (less colors) than JP and Greedy
- general exponential correlation between number of colors and number of nodes
- time advantages of parallelism become more evident with big graphs

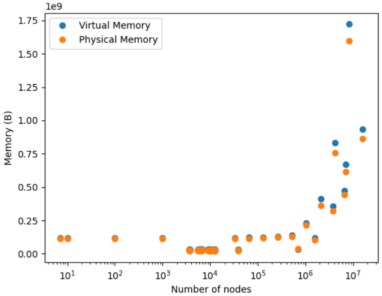
Results - 2





- parallel implementations of the same algorithms are faster, especially for big graphs
- standard and *find and color* implementations are similar. Standard is better for bigger graphs, *find and color* for smaller ones





- Using a threadpool provides a relevant advantage, especially for small graphs
- Memory occupancy grows significantly for big graphs