Luby’s Algorithm for finding Maximal Independent Set

15418 Final Project - Spring 2022

Clarissa Xu and Arvind Mahankali

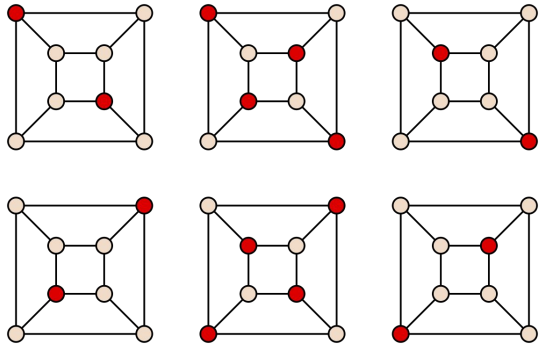
5th May, 2022

Agenda

- Background
- Luby's Algorithm Overview
- What We Did
- Our Earlier Approaches
- Our Approach
- Runtimes and Speedup

Background

- Input: Graph G = (V,E) in adjacency list representation
- Output: Maximal Independent Set
  - A set S of vertices in G such that there is no edge in G between any 2 vertices in S, and for any vertex v not in S, there is an edge from some vertex in S to v.



Luby's Algorithm

1. Determine if there are any vertices left in the active set A
2. Assign random priorities to all of the vertices in the active set
3. For each vertex v in A, go through the priorities of all neighbors and find the max of those. Compare this to v's priority and determine if v should be added to the independent set
4. Update the independent set and remove the newly added vertices from the active set

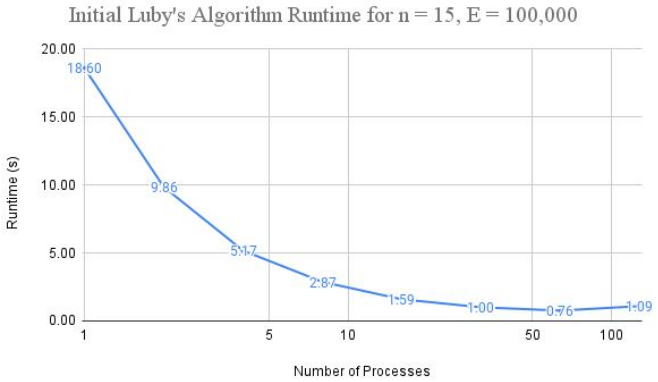
What we did

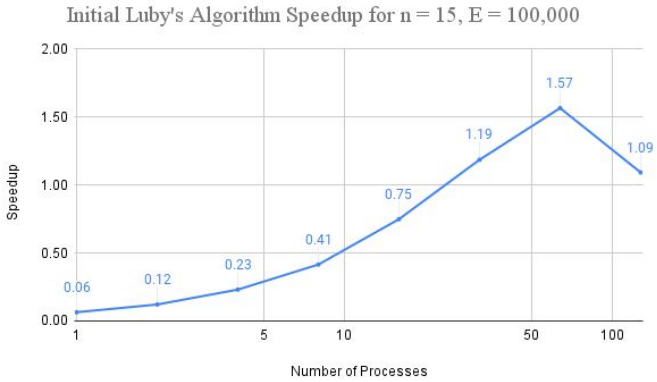
- Greedy sequential algorithm
- Sequential version of Luby's algorithm
- 3 parallel programs with MPI
- 1 program to generate random undirected graphs for testing
- 1 checker python program to check if outputted a valid maximal independent set

V1: Initial Luby's

- Root determines all the priorities and all the active sets and broadcasts them
- Each process determines which of their vertices are in the independent set

V1: Runtimes and Speedup

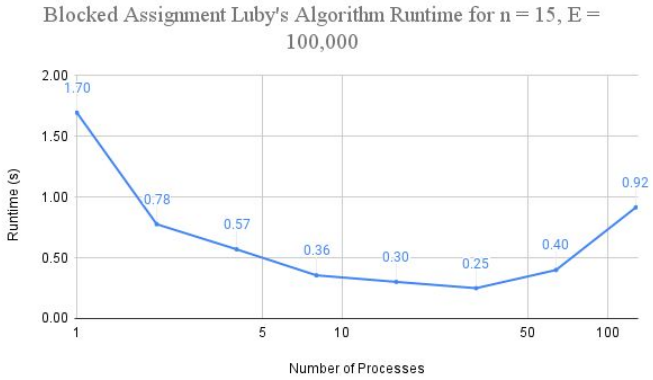


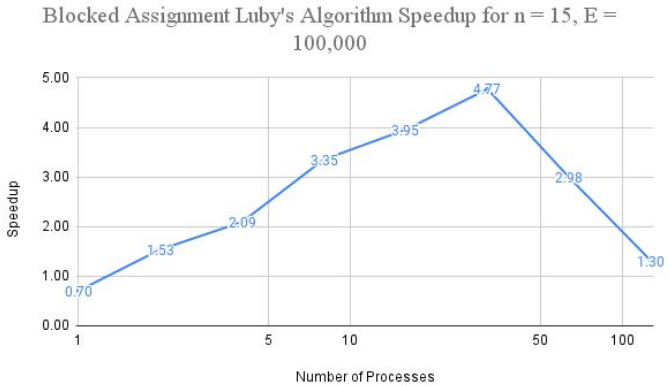


V2: Luby's with Blocked Assignment

- Each process calculates priorities for their vertices
- Each process sends the priority of a vertex to each of their neighbors
  - Synchronous Send and Receive Calls
- Each process p sends a message to every other process q containing neighbors of vertices in q that are no longer in the active set

V2: Runtimes and Speedup

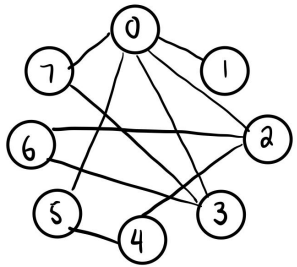




V3: Luby's with Blocked Assignment and Single Messaging

- We collate communication to send all pertinent priority and active set neighbor information in one message to each process, rather than by edge
- We observe that a process p does not have to exchange messages with process q if they don't have edges between active vertices

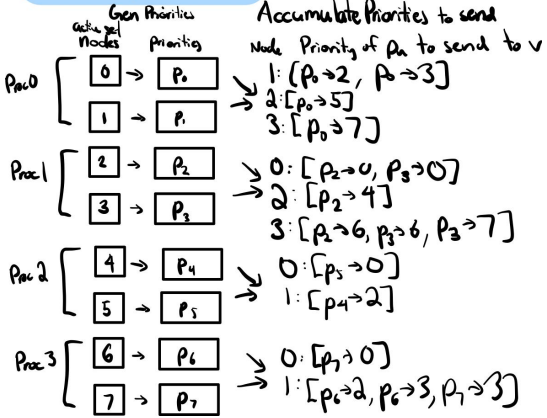
Initialization



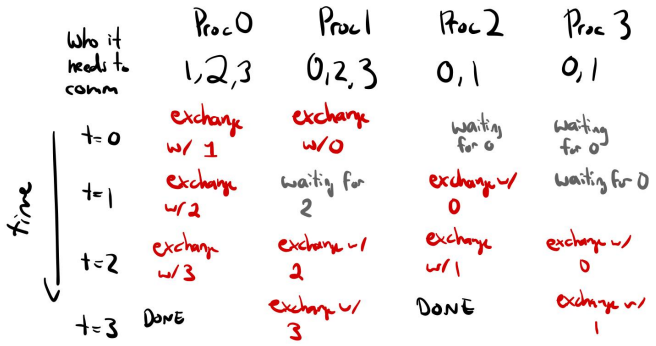
Adjacency list

0 : [1,2,3,5,7]
1 : [0]
2 : [0,4,6]
3 : [0,6,7]
4 : [2,5]
5 : [0,4]
6 : [2,3]
7 : [0,3]

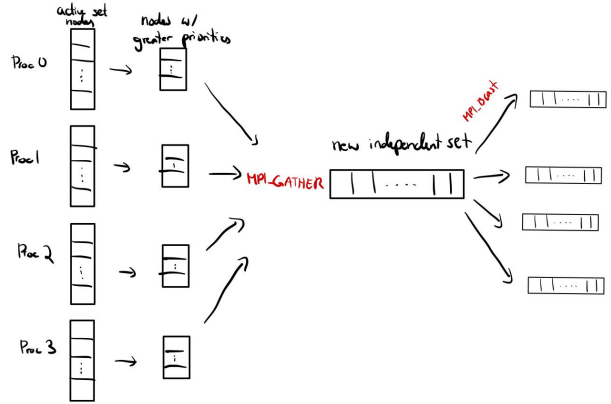
Priorities Accumulation



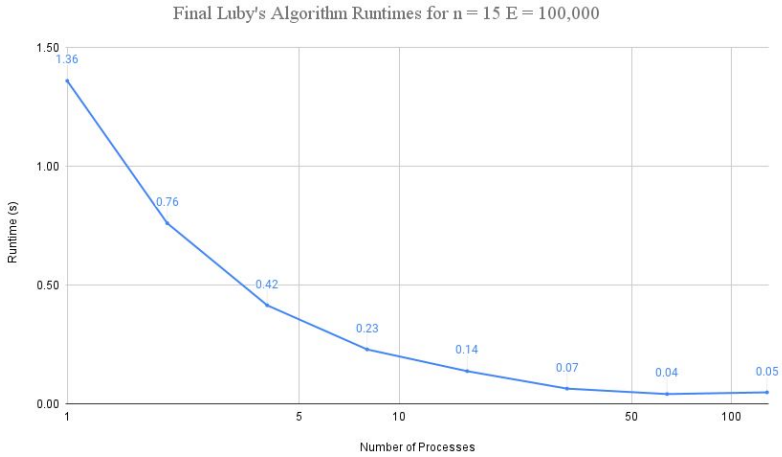
Sending of Priorities

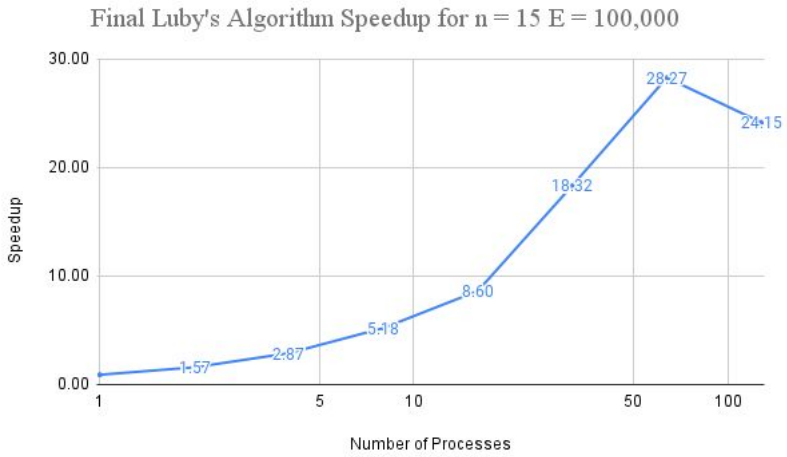


Determining new independent set

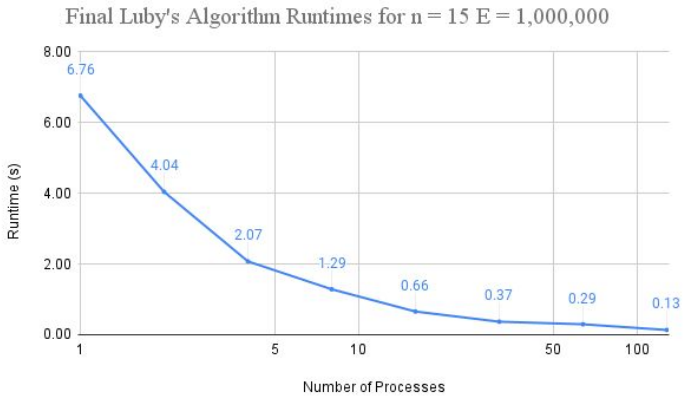


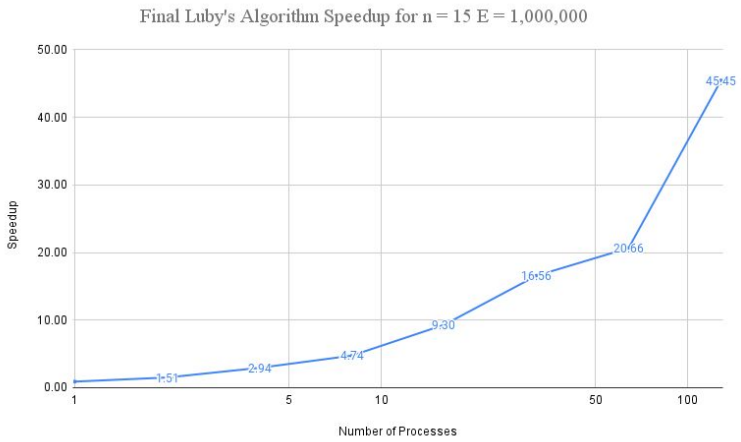
V3: Runtimes and Speedup





V3: Runtimes and Speedup





V3: Runtimes and Speedup