PARALLEL IMPLEMENTATION OF BELLMAN FORD ALGORITHM

CSE 633 – Parallel Algorithms

Instructor : Dr. Russ Miller

Presented by Shreya Reddy Gouru

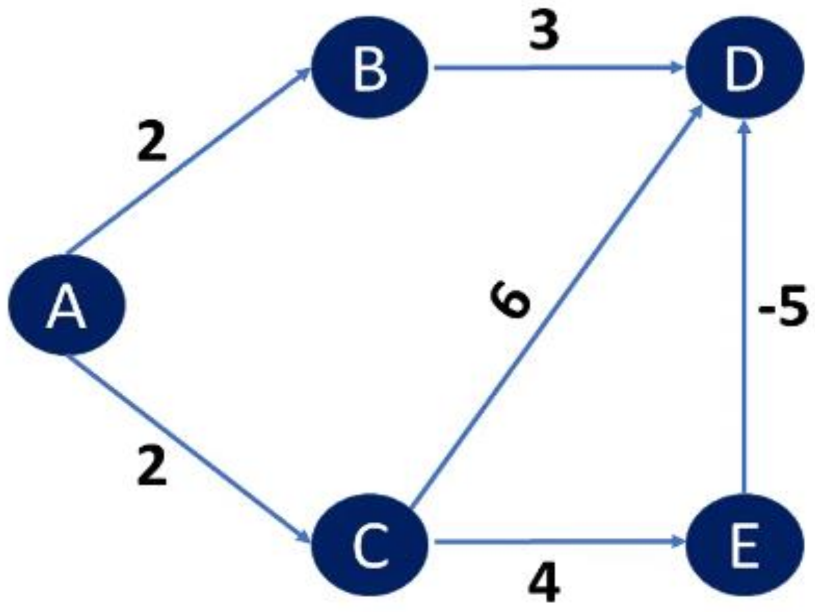
University at Buffalo The State University of New York

Outline

- Problem Statement
- Bellman Ford Algorithm
- Example
- Sequential Algorithm
- Approaches to Parallelize it
- Pseudo code for Course grained approach
- To do task status
- Implementation Results
- Future work
- References

Problem Statement

- Single source shortest path.
- To find shortest path from a given vertex to all other vertices in a weighted directed graph.
- To detect negative cycles in the graph



Bellman Ford Algorithm

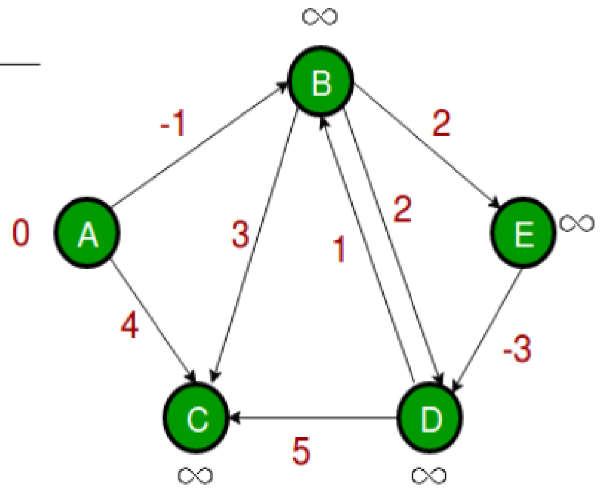
- Computes shortest path from a source to all vertices in a weighted graph.
- Capable of handling graphs with negative edge weights.
- Dijkstra vs Bellman Ford.
- Applications in routing.

How it works?

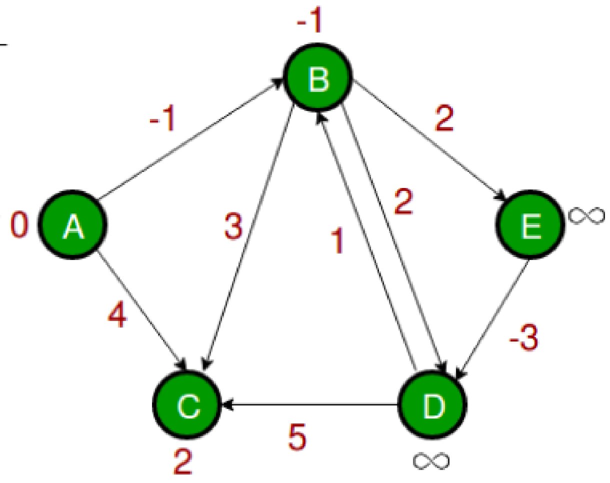
- Relaxes all edges |V-1| times to approximate distances, where |V| is the number of vertices in a graph.
- Incase of negative cycle, distances are updated even after last iteration.

Example

A	B	C	D	E
0	∞	∞	∞	∞

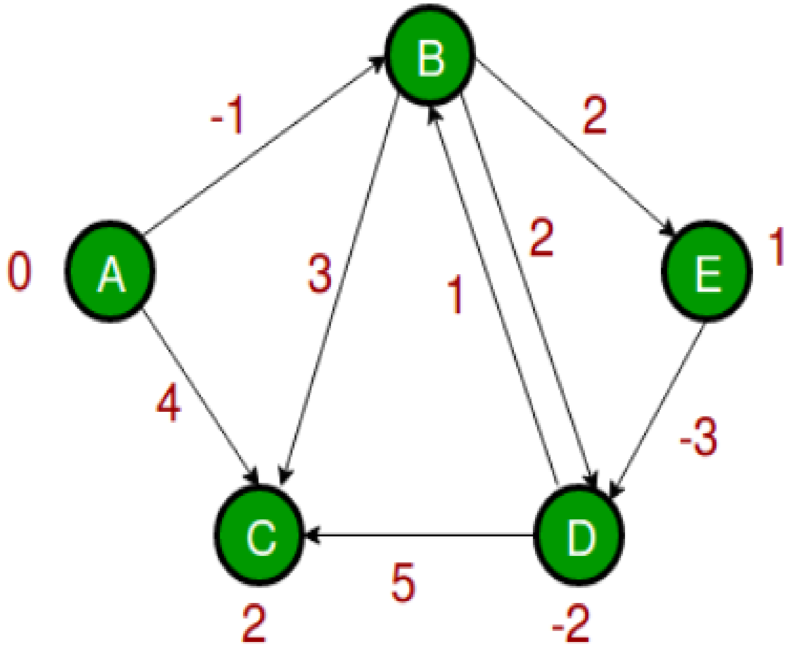


A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞



Example

A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	∞	1
0	-1	2	1	1
0	-1	2	-2	1



Sequential Algorithm

```
function bellmanFord(G, S)
  for each vertex V in G
    distance[V] <- infinite
      previous[V] <- NULL
  distance[S] <- 0

  for each vertex V in G
    for each edge (U,V) in G
      tempDistance <- distance[U] + edge_weight(U, V)
      if tempDistance < distance[V]
        distance[V] <- tempDistance
        previous[V] <- U

  for each edge (U,V) in G
    If distance[U] + edge_weight(U, V) < distance[V}
      Error: Negative Cycle Exists

  return distance[], previous[]
```

TIME COMPLEXITY : O(V*E)

Approaches to Parallelize it

COARSE GRAIN

- Each Processor is assigned a subset of edges in the beginning and assignment never changes.
- Iteratively performs computation and communication phases.
- Each processor relaxes its subset of edges and updated local distances.
- At the end of computation, distance vector equal to the minimum of all labels is updated in all processors.

FINE GRAIN

- Each Processor maintains a list of vertices ordered by the labels in the distance vector
- During communication phase, each processor selects minimum element on its local distance vector and a vertex which has least distance is selected by all processors.
- Edges from that vertex are relaxed by all processors in its subgraph. Computation phase is same in both approaches.

Pseudocode for coarse grained Algorithm

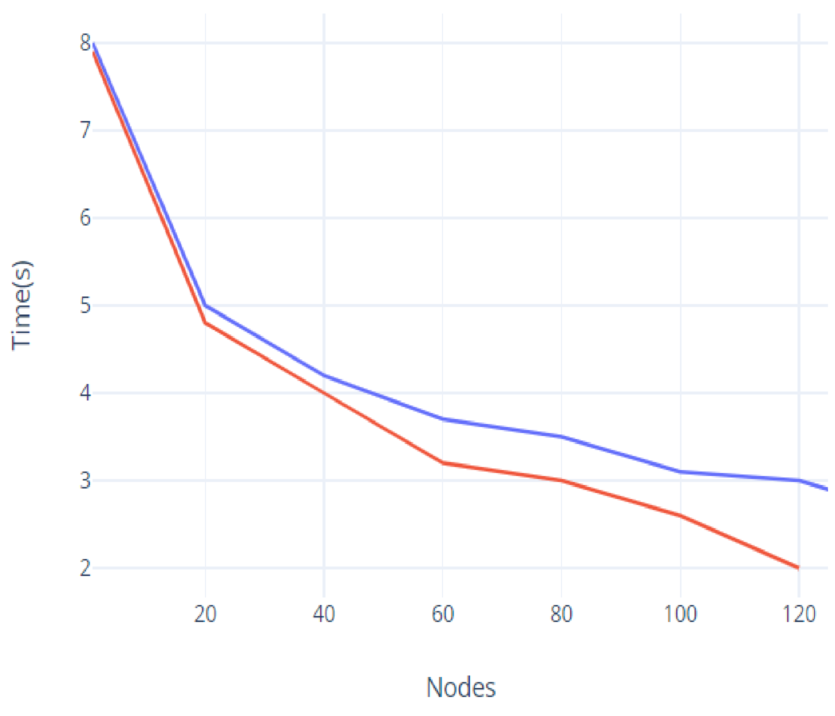
```
f_g = f_l                    ;;; f_l is initially FALSE
f_l = FALSE
for each vertex u
    do if d(u) > d_min(u)
        then d(u) ← d_min(u)
             π(v) ← ∞
             f_g = TRUE
             if outdegree(u) > 0
                then mark u
for each vertex u in order
    do if u is marked
        then unmark u
             for each edge (u, v)
                do if d(v) < d(u) + w(u, v)
                    then d(v) ← d(u) + w(u, v)
                         π(v) ← u
                         f_l = TRUE
                         if outdegree(v) > 0
                            then mark v
if f_g = FALSE
    then terminate
```

To do

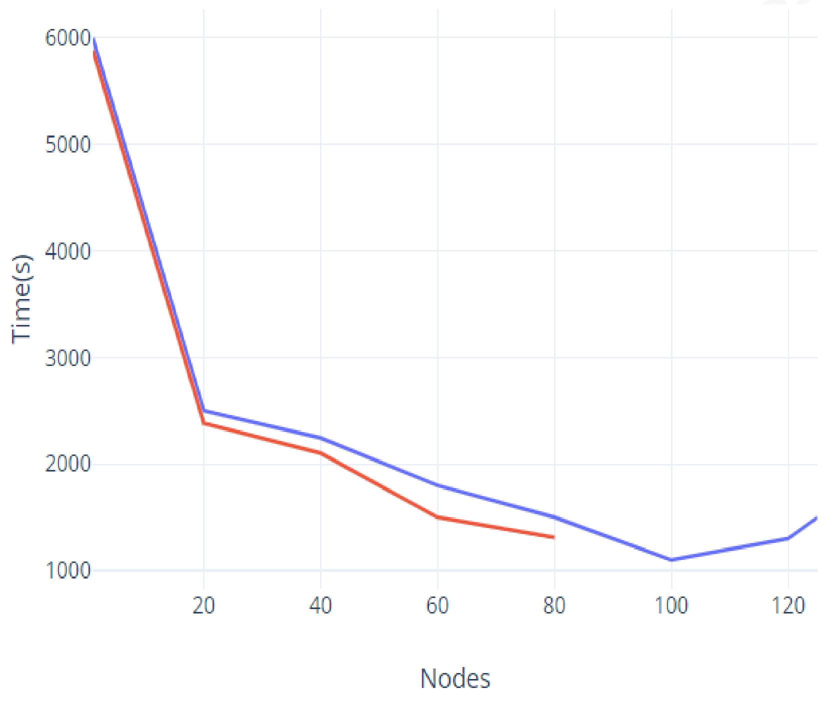
- Run the algorithm on larger input ✓
- Implement couple of heuristics from research paper ✓
- Fine grained approach ✓
- Comparison of Course grained and Fine grained approach
- Negative cycle detection ✓

Implementation Results

GRAPH : 1000 VERTICES

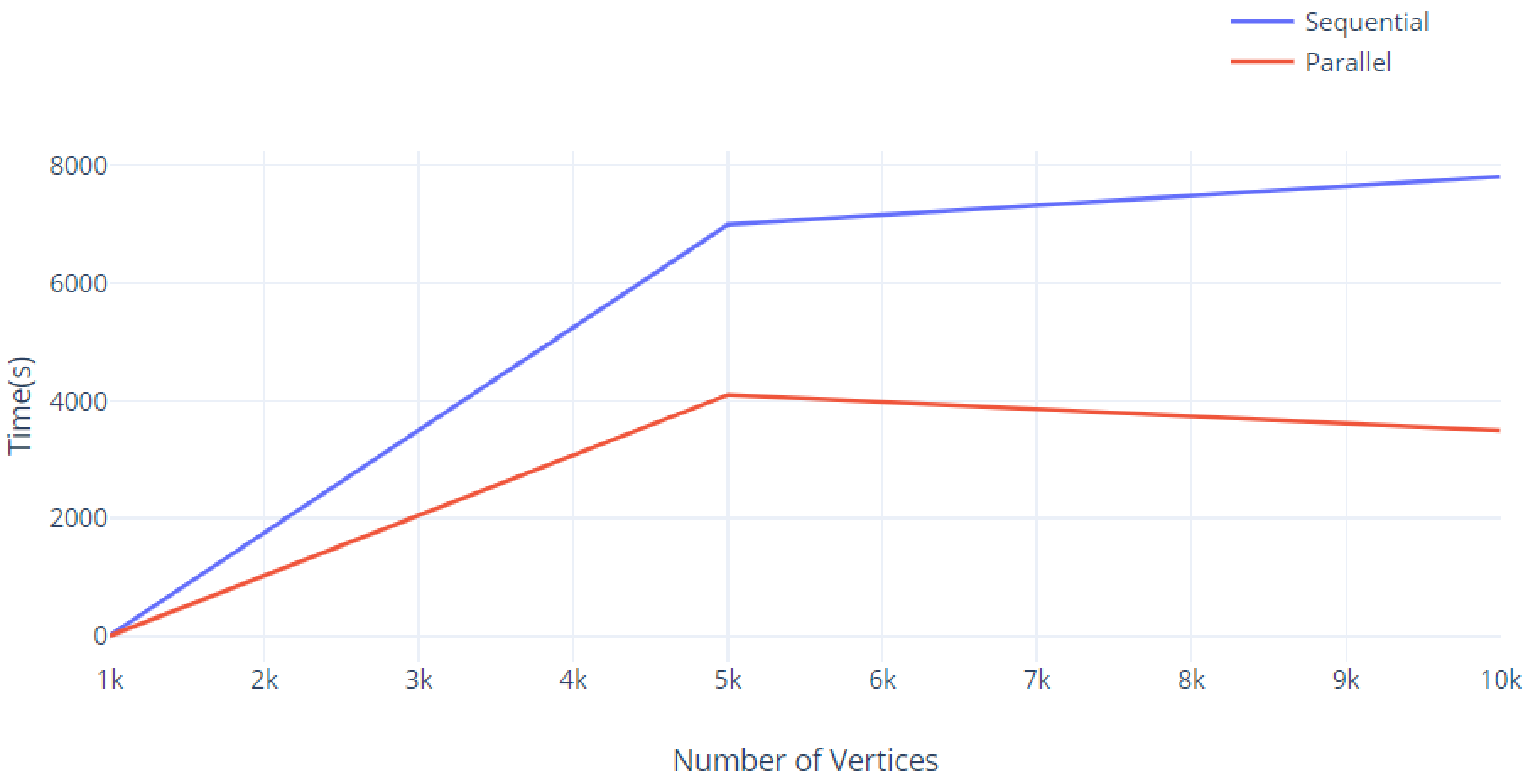


GRAPH : 5000 VERTICES



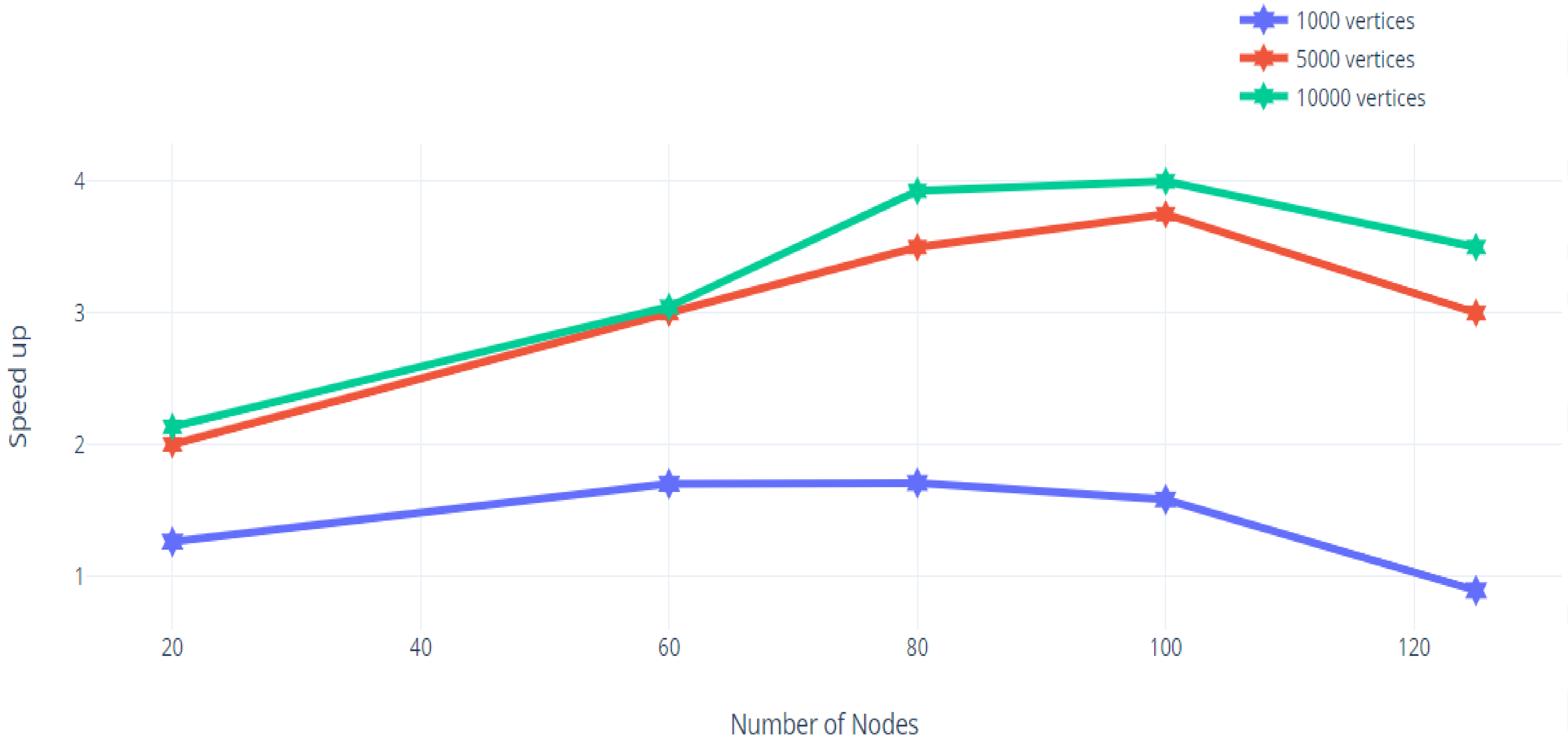
GRAPH : 10000 VERTICES

Sequential vs Parallel



Speed up

Speed Up = $T_{sequential} / T_{parallel}$



Future Work

- Implementation using CUDA
- Fine grained approach on large number of nodes
- Increase input data up to 2^{32} vertices

References

- Implementing Parallel Shortest-Paths Algorithms (1994) by Marios Papaefthymiou and Joseph Rodrigue.
- Y. Tang, Y. Zhang, H. Chen, “A Parallel Shortest Path Algorithm Based on GraphPartitioning and Iterative Correcting”, in Proc. of IEEE HPCC’08, pp. 155-161, 2008.
- https://cse.buffalo.edu/faculty/miller/Courses/CSE633/Muthuraman-Spring-2014-CSE633.pdf.pdf
- Algorithms Sequential & Parallel: A Unified Approach by Russ Miller and Lawrence Boxer
- https://mpitutorial.com/tutorials/
- https://www.programiz.com/dsa/bellman-ford-algorithm
- https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/

Thank You