OPTIMAL DETERMINISITC MASSIVELY PARALLEL CONNECTIVITY ON FORESTS

Presented by Maarij Imam and Ayaan Ahmed

INTRODUCTION

- **OBJECTIVE**: SOLVE THE MAX-ID PROBLEM ON TREES IN THE MASSIVELY PARALLEL COMPUTATION (MPC) MODEL.
- **MAX-ID PROBLEM**: GIVEN A TREE WITH UNIQUE NODE IDS, EACH NODE OUTPUTS THE MAXIMUM ID IN THE TREE.
- **WHY IMPORTANT?:** CORE COMPONENT FOR SOLVING CONNECTED COMPONENTS IN FORESTS.
- **PAPER'S CONTRIBUTION:** SOLVES MAX-ID IN O(LOG D) ROUNDS, WHERE D IS THE TREE'S DIAMETER.

MPC MODEL

THE MASSIVELY PARALLEL COMPUTATION (MPC) MODEL IS A FRAMEWORK FOR PARALLEL COMPUTING WHERE MULTIPLE MACHINES HAVE:

- **LOW-SPACE REGIME:** EACH MACHINE HAS MEMORY PROPORTIONAL TO N^{Δ}, WHERE $0 < \Delta < 1$.
- **TOTAL MEMORY:** PROPORTIONAL TO N + M, WHERE N IS NODES, M IS EDGES.
- **SYNCHRONOUS ROUNDS** AND **COMMUNICATION B/W MACHINES**

MAX-ID SOLVER OVERVIEW

- **GOAL**: EVERY NODE LEARNS THE MAXIMUM ID IN THE TREE.

- **HIGH-LEVEL APPROACH:**
- COMPRESS GRAPH (LIGHT SUBTREES AND PATHS) TO A SINGLE NODE HOLDING THE MAXIMUM ID.
- DECOMPRESS TO BROADCAST THE MAXIMUM ID TO ALL NODES.

- **ALGORITHM STRUCTURE:**
- CONSTANT NUMBER OF COMPRESSION PHASES (COMPRESSLIGHTSUBTREES, COMPRESSPATHS).
- REVERSAL PHASES (DECOMPRESSION) TO RESTORE GRAPH AND SPREAD MAXIMUM ID.

- **KEY DEFINITIONS:**
- **LIGHT NODE:** NODE V IS LIGHT AGAINST NEIGHBOR U IF SUBTREE HAS AT MOST $N^{(\Delta/8)}$ NODES.
- **HEAVY NODE:** NODE THAT IS NOT LIGHT.
- **NODE STATES:** ACTIVE, HAPPY, FULL, SAD (FOR COMPRESSION MANAGEMENT).

MAX-ID-Solver$(G, \hat{D})$

Initialize $G_0 \leftarrow G$

1. For $i = 0, \ldots, \ell - 1$ phases:
   (a) $G'_i = \mathsf{CompressLightSubTrees}(G_i, \hat{D})$
   // If there are heavy nodes, all light nodes are compressed into the closest heavy node. Otherwise, all nodes are light and are compressed into a single node.
   (b) $G_{i+1} = \mathsf{CompressPaths}(G'_i, \hat{D})$
   // All paths are compressed into single edges.
2. For $i = \ell - 1, \ldots, 0$ reversal phases:
   (a) $G'_i = \mathsf{DecompressPaths}(G_{i+1})$
   // All paths that were compressed during Step 1(b) are decompressed.
   (b) $G_i = \mathsf{DecompressLightSubTrees}(G'_i)$
   // All light nodes that were compressed during Step 1(a) are decompressed from v.

COMPRESSION PROCEDURES

COMPRESSING LIGHT SUBTREES

- **PURPOSE**: COMPRESS LIGHT SUBTREES INTO ADJACENT HEAVY NODES.
- **MECHANISM:** NODES MAINTAIN SET S_V (INITIALLY NEIGHBORS). USE CAREFUL **EXPONENTIATION** TO GROW S_V. NODES BECOME HAPPY IF THEY LEARN A LIGHT SUBTREE (**SIZE AT MOST N^(Δ/8)**).
- **PROBING:** ESTIMATE SUBTREE SIZES TO AVOID HEAVY PARTS.
- **OUTCOME**: LIGHT NODES BECOME HAPPY AFTER O(LOG D) ITERATIONS (LEMMA 4.20, PAGE 12).

COMPRESSION PROCEDURES

COMPRESSING PATHS

- **PURPOSE**: REPLACE ALL PATHS IN THE GRAPH WITH SINGLE EDGES.
- **MECHANISM**: AFTER COMPRESSING LIGHT SUBTREES, CONTRACT PATHS (**SEQUENCES OF DEGREE-2 NODES**) INTO EDGES, PRESERVING CONNECTIVITY AND MAXIMUM ID KNOWLEDGE.
- **IMPACT**: REDUCES GRAPH SIZE, ENSURES NO DEGREE-2 NODES REMAIN, RUNS IN O(1) ROUNDS PER PHASE.

DECOMPRESSING TO BROADCAST THE MAXIMUM ID

- **PURPOSE**: REVERSE COMPRESSION TO RESTORE ORIGINAL GRAPH AND SPREAD MAXIMUM ID.
- **MECHANISM:**
- **DECOMPRESSPATHS:** EXPAND EDGES BACK TO PATHS, ASSIGN MAXIMUM ID TO RESTORED NODES.
- **DECOMPRESSLIGHTSUBTREES:** EXPAND SUBTREES, PROPAGATE MAXIMUM ID TO ALL NODES.
- RUNS IN O(LOG D) ROUNDS, SAME AS COMPRESSION.

COMPLEXITY

TIME COMPLEXITY:

- RUNS IN O(LOG D) ROUNDS, WHERE D IS IN [DIAMETER(G), N^(DELTA/8)] (LEMMA 4.2, PAGE 10).
- L = O(1) PHASES, SET BY CONSTANT GRAPH SIZE REDUCTION (LEMMA 4.14, PAGE 51).
- EACH COMPRESSION/DECOMPRESSION PHASE TAKES O(LOG D) ROUNDS.

MEMORY COMPLEXITY:

- **GLOBAL MEMORY**: O(N * D^3) FOR MAX-ID (LEMMA 4.24, PAGE 13).REDUCED TO O(N + M) WITH PREPROCESSING (SECTION 5, PAGE 8).
- **LOCAL MEMORY**: O(N^Δ) PER MACHINE, RESPECTING LOW-SPACE MPC

SUMMARY

- MAX-ID solver computes maximum ID in O(log D) rounds using compression (CompressLightSubTrees, CompressPaths) and decompression.
- Decompression ensures all nodes learn maximum ID, enabling connected components solution in O(log D) rounds with O(n + m) memory.
- Significance:
- Removes n dependency, improves prior O(log D + log log n) results.
- Optimal under 1 vs. 2 cycles conjecture.

Thank You