Optimized Parallel Breadth-First Search With Adaptive Strategies

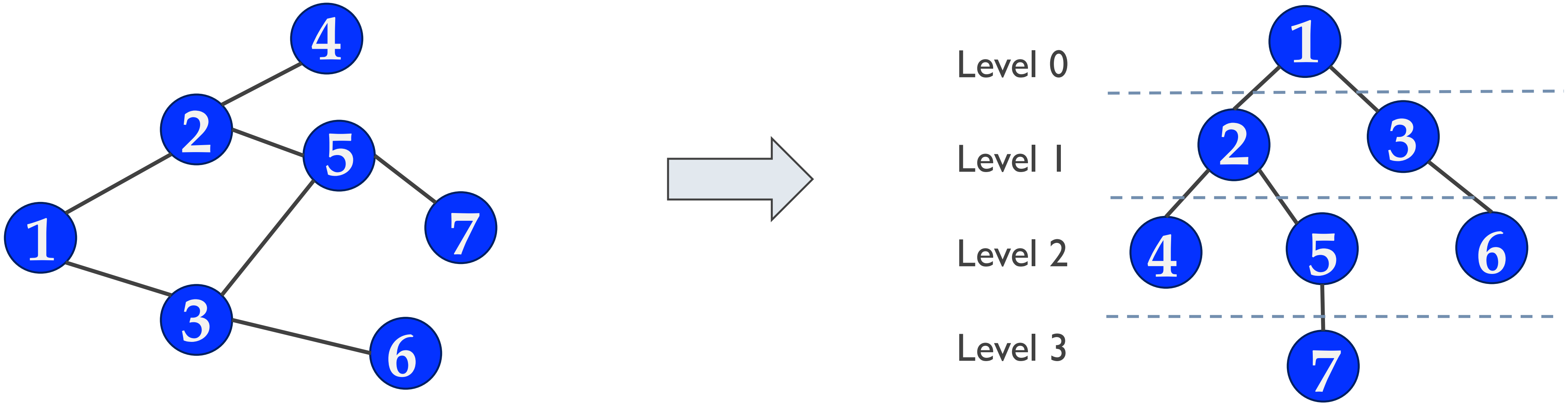
Chaoqun Li Runbang Hu Xiaojiang Du Yuede Ji

UNIVERSITY OF TEXAS ARLINGTON
UNIVERSITY OF TEXAS ARLINGTON
STEVENS INSTITUTE OF TECHNOLOGY 1870
UNIVERSITY OF TEXAS ARLINGTON

UNIVERSITY OF TEXAS ARLINGTON

What is BFS

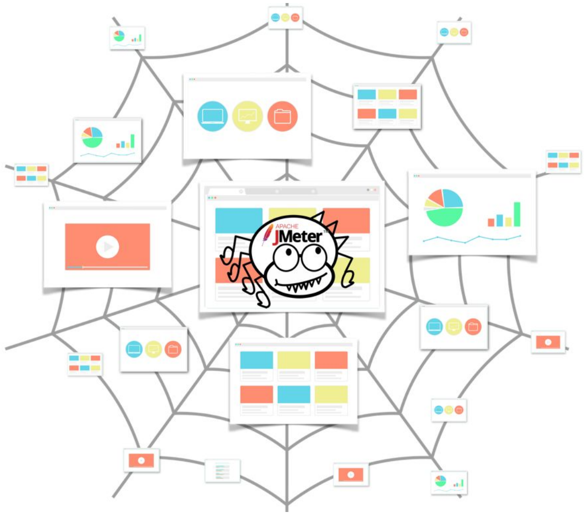
- Breadth-First Search (BFS) is a fundamental graph traversal algorithm using a level-by-level pattern.
- Time complexity: O(|V| + |E|)



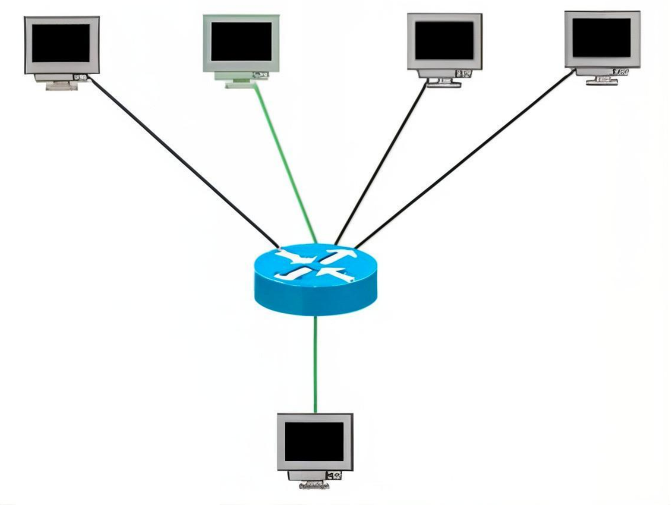
Importance of BFS

- BFS can be used in…

Social Networks

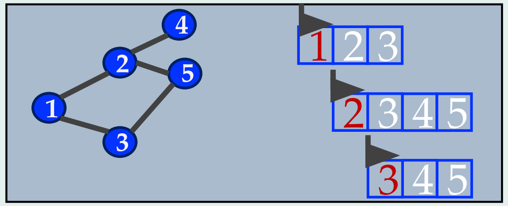
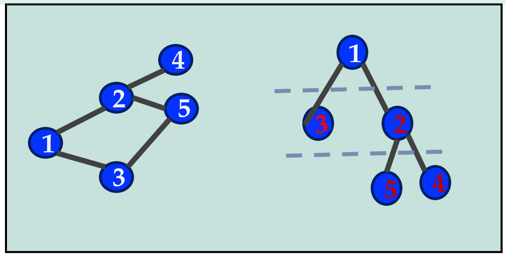
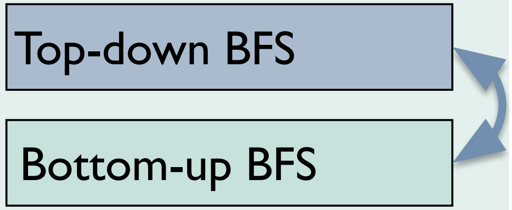


Web Crawling



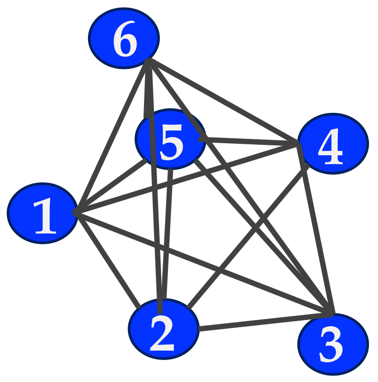
Network Broadcasting

BFS Algorithm

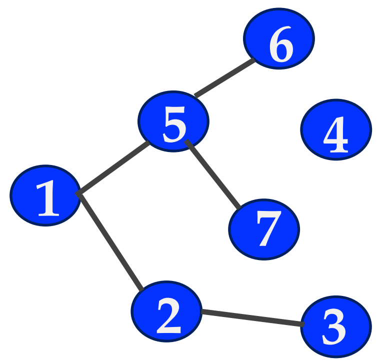
Algorithm	Description	Flow
Top-Down BFS	multiple threads process nodes in parallel. all frontier nodes are explored at the same time	
Bottom-Up BFS	unvisited nodes actively check if any of their neighbors belong to the previous level	
Hybrid BFS	dynamically switches between top-down and bottom-up strategies depending on the size of the frontier	

Challenge #1:Graph Diversity

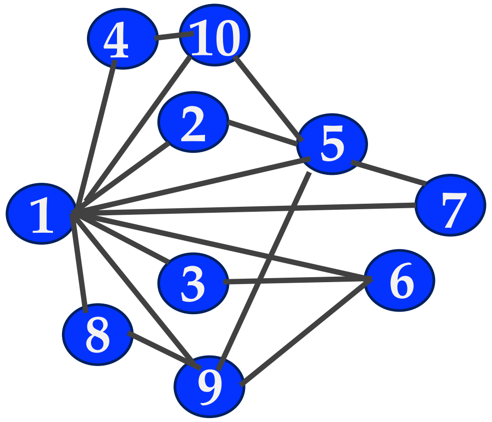
- Dense graph: High average degree



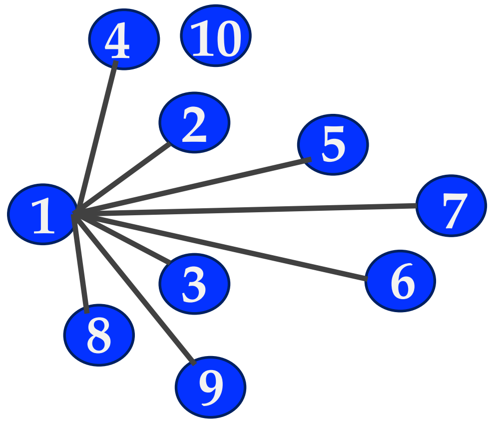
- Sparse graph: Few edges compared to the number of nodes



- Power-law graph: High-degree nodes dominate

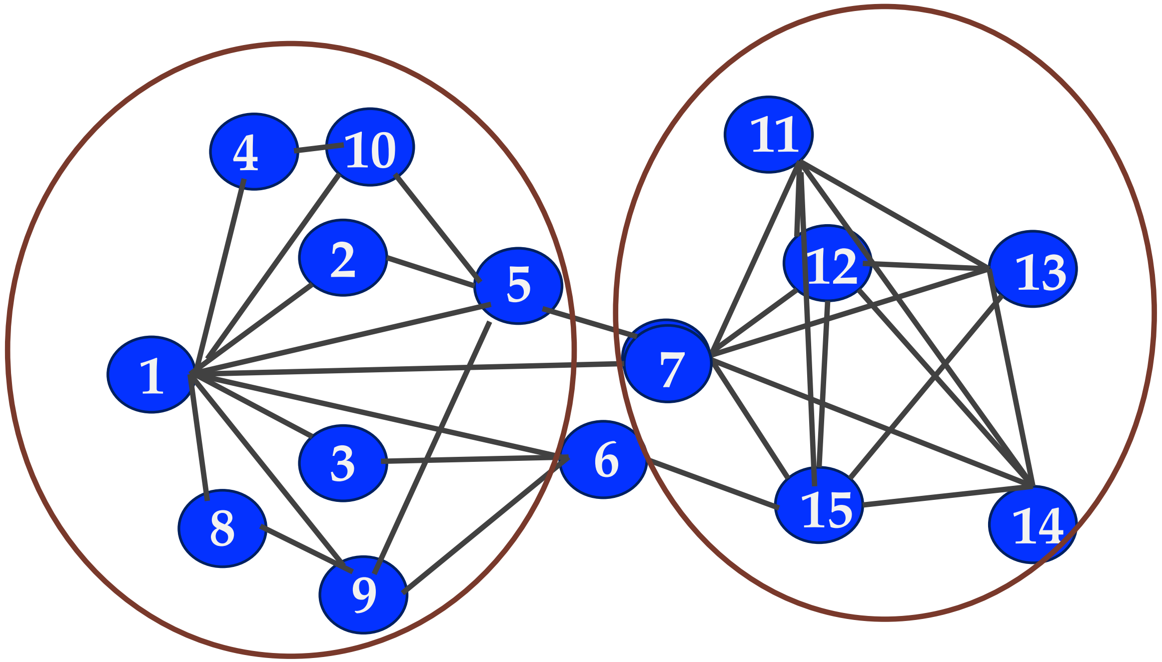


- High-degree skewed graph: Extreme degree imbalance



Challenge #1: Graph Diversity

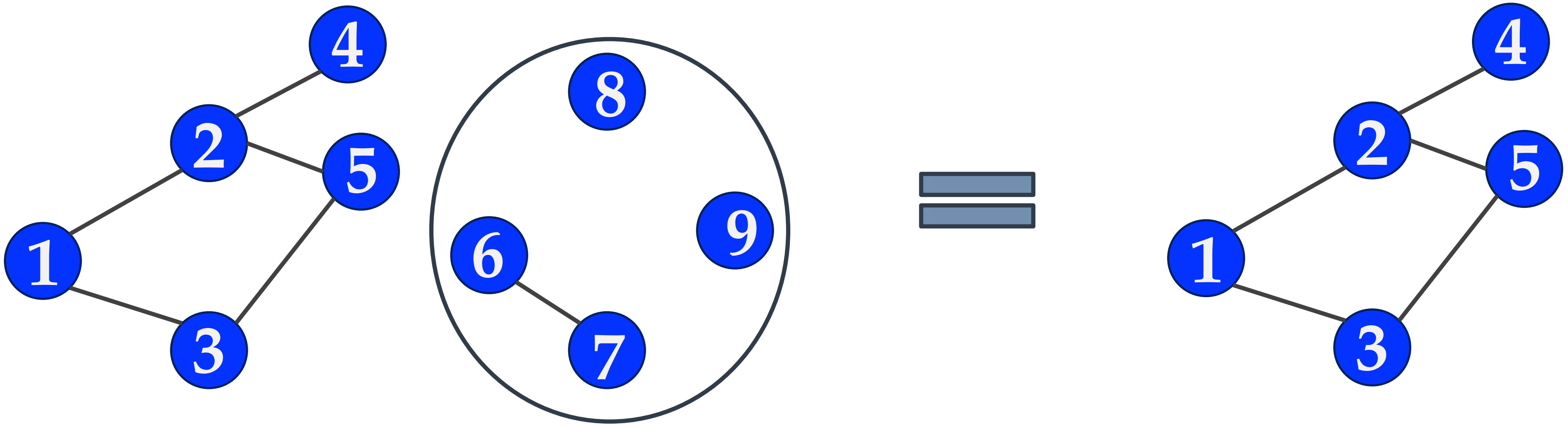
- Some graphs have multiple properties in different parts



No single strategy fits all

Challenge #2: Redundant Computation

For vertices with a degree of 1 or 0, they should not be visited or searched because they do not contribute to the connectivity of the graph.



If source isn't included

Technique #1: Graph-Type Aware

- Classification of graphs using a 4-bit mask:
  - Grant graph properties based on their differences

Graph-properties (binary)	Cache-friendly	Sharp graph	Considered dense	High degree graph
1000	Yes	No	Yes	No
0100	No	Yes	No	No
0010	No	No	Yes	No
0001	No	No	No	Yes

Cache-friendly: 8*N+4*M < 32KB
Sharp: Max_degree> 128
Dense: Average_degree >= 8
High degree: max_degree / average_degree >= 16

Technique #1: Graph-Type Aware

Graph	Definitions	Values	Example
Power-law graph	High-degree nodes dominate	0b0111	Web_Graph_1
High-degree skewed graph	Extreme degree imbalance	0b0101	KNN_Graph_1
Dense graph	High average degree	0b0011	Synth_Dense_1
Cache-fitting graph	Fits in L1 cache	0b1010	Rand_1k_5k

Technique #1: Graph-Type Aware

- Power-law graph & High-degree skewed graph

Dynamic Switching

Top-down BFS

Bottom-up BFS

```
while (!queue.empty()) {
    if (scout_count > top_threshold && is_dense){
        // Switch to bottom-up
        QueueToBitmap(queue, front);
        queue.slide_window();

        int64_t awake_count = 0, old_awake_count;
        do {
            old_awake_count = awake_count;
            awake_count = BUStep(front, next, distances);
            front.swap(next);
        } while (awake_count >= old_awake_count || awake_count > bottom_threshold);

        BitmapToQueue(front, queue);
        scout_count = 1;
    } else {
        // Top-down step
        //edges_to_check -= scout_count;
        scout_count = TDStep(queue, distances);
        queue.slide_window();
    }
}
```

Power-law graph
Early-in late out

High-degree skewed graph
Early-in early out

Frontier/unvisited nodes >Threshold

Technique #1: Graph-Type Aware

- Dense graph
- Cache-fitting graph

Dynamic Switching

Top-down BFS

Bottom-up BFS

Top-down □ Bottom-up

Dynamic Switching

Top-down BFS

Two-level Bottom-up BFS

Top-down

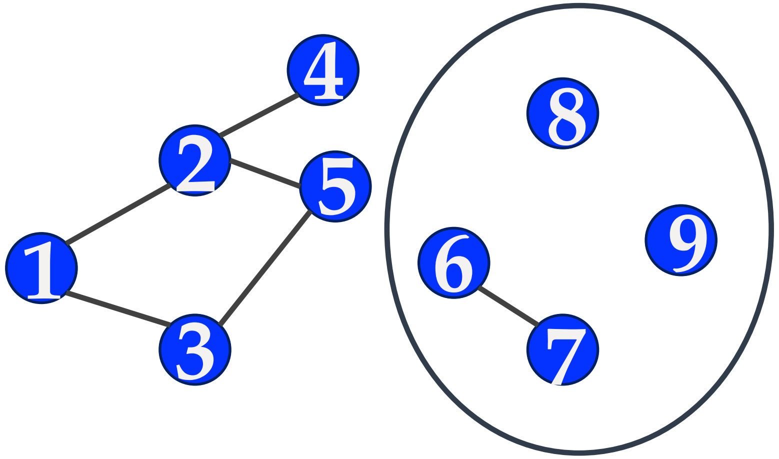
□Two-level Bottom-up

```
for (vidType v = 0; v < N; ++v) {
    if (distances[v] == std::numeric_limits<weight_type>::max()) {
        bool is_early_termination = false;
        bool has_cur_neighbor = false;
        eidType start = rowptr[v];
        eidType end = rowptr[v + 1];
        for (eidType edge_idx = start; edge_idx < end; ++edge_idx) {
            vidType neighbor = col[edge_idx];
            if(distances[neighbor] == level) {
                    distances[v] = level + 1;
                    if(is_changed_before == false) {
                        is_changed_before = true;
                    }
                    is_early_termination = true;
                break;
            } else if(distances[neighbor] == level + 1) {
                has_cur_neighbor = true;
            }
        }
    if(!is_early_termination && has_cur_neighbor) {
        distances[v] = level+2;
        is_changed_before = true;
    }
```

Technique #2: Graph Pruning

- These vertices form connected components of size-1 (isolated vertices) or size-2 (vertices connected by a single edge).

- Prunes size-1 and size-2 connected components (CCs)



Size-1 CC: 8, 9
Size-2 CC: 6, 7

Graph	Pruned Nodes	Original Nodes
Collaboration_Network	98	1,058,365
Road_Network_1	64,214	21,872,120
Road_Network_2	322,266	86,081,964
Social_Network_1	2,316	21,872,120

Other Optimizations

- *Sliding queue and bitmap*
  - Using more efficient data structures to enhance parallelization and implemented a bitmap to reduce computational workload.
  - Benefit: Significant overall speedup, particularly in large, dense graphs.

A. Azad et al., "Evaluation of Graph Analytics Frameworks Using the GAP Benchmark Suite," 2020 IEEE International Symposium on Workload Characterization (IISWC), Beijing, China, 2020, pp. 216-227, doi: 10.1109/IISWC50251.2020.00029.

Experimental Setup

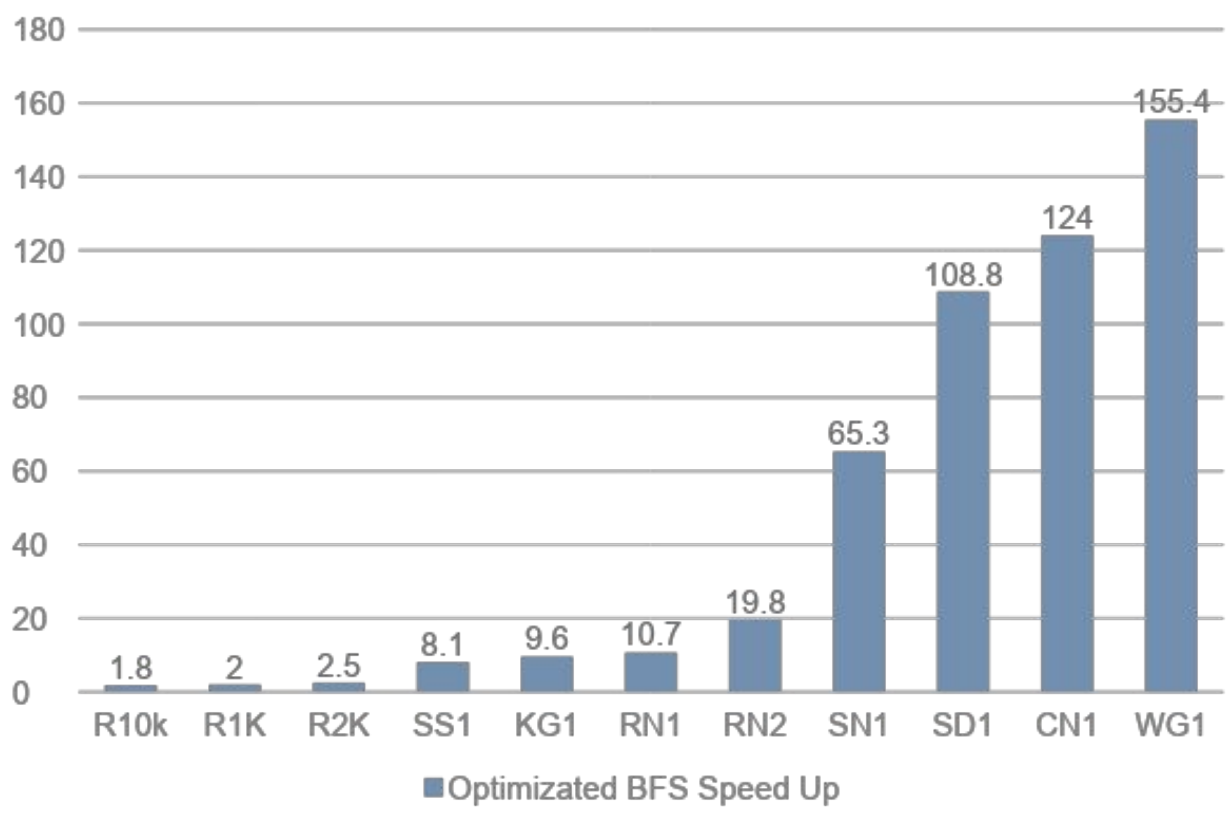
- Implementation
  - 550+ lines of code in C++
  - Include graph preprocessing
- Environments
  - CPU cores: 24 cores for parallel execution
  - Memory: 96 GB of RAM

Graph (Abbr.)	|Vertex|	|Edge|	Avg. degree
Road_Net_2	86.08M	216M	2
KNN_Graph_1	24.63M	154M	6
Road_Net_1	21.87M	58M	2
Synth_Dense_1	9.90M	980M	98
Synth_Sparse_1	9.90M	39M	3
Web_Graph_1	6.56M	294M	44
Social_Net_1	4.8M	84M	17
Collab_Net_1	1.06M	110M	104
Road_2k_10k	2K	19.94K	9
Road_1k_5k	1K	9.94K	9
Rand_10k_50k	10K	100K	9

Performance Comparison

- Overall speedups:
- Speedup = Single-threaded runtime / Our adaptive strategies

  - average speedup of **9.5×**
  - peak of **155×** for web graphs
  - Up to **1.89** Billion edges per second



Thank You!

- Email:
  - cxl6029@mavs.uta.edu
  - rxh0841@mavs.uta.edu
  - xdu16@stevens.edu
  - yuede.ji@uta.edu

- Webpage:
  - https://keparal.cn
  - https://www.stevens.edu/profile/xdu16
  - https://yuede.github.io/