

Direction-Optimizing Breadth-First Search



Scott Beamer · Krste Asanović · David Patterson

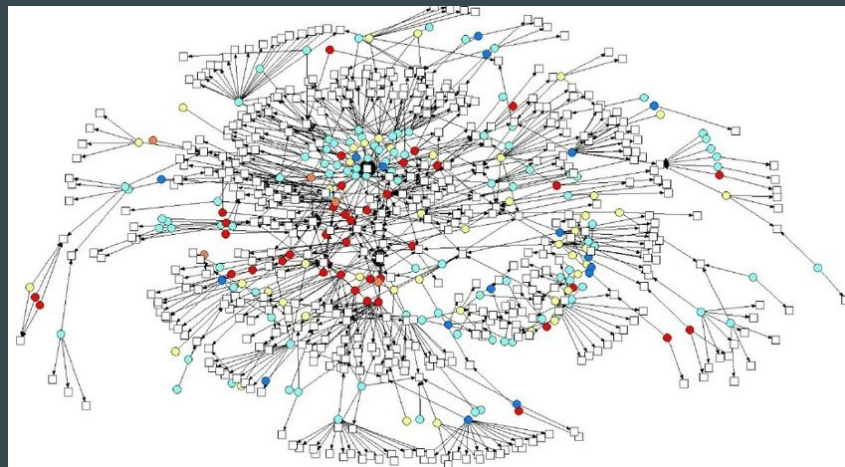
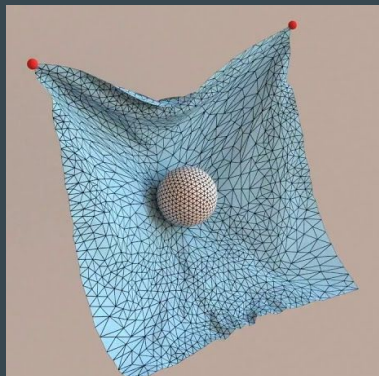
Presented by Abrianna Zhang
6.5060 · 9/17/2027

Introduction

- Graph algorithms are often slow due to lack of memory locality (BFS)
- Paper presents BFS algorithm
 - combines top-down with bottom-up approach
 - significant speedups on state-of-the-art

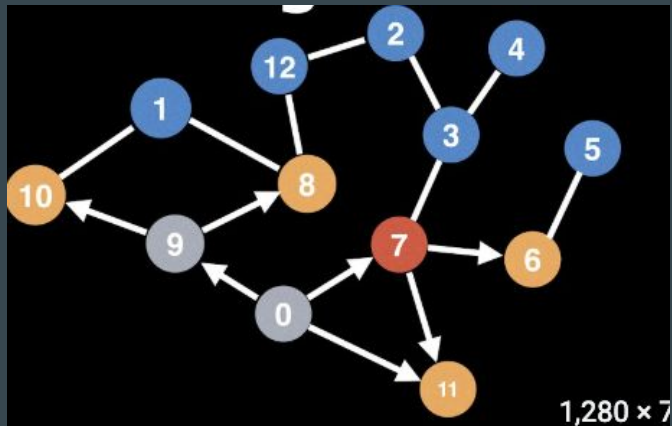
Graph Properties

- Characterize by **diameter** and **degree distribution**
 - mesh graphs: high diameter, low degree -> partition
 - *social networks*: **small world** and **scale-free**
 - low diameter
 - few, high-degree nodes



Conventional Top-Down BFS Algorithm

- Start from source vertex
- Frontier expands
- Visit vertices at same depth
 - Check neighbors
 - Add unvisited to frontier

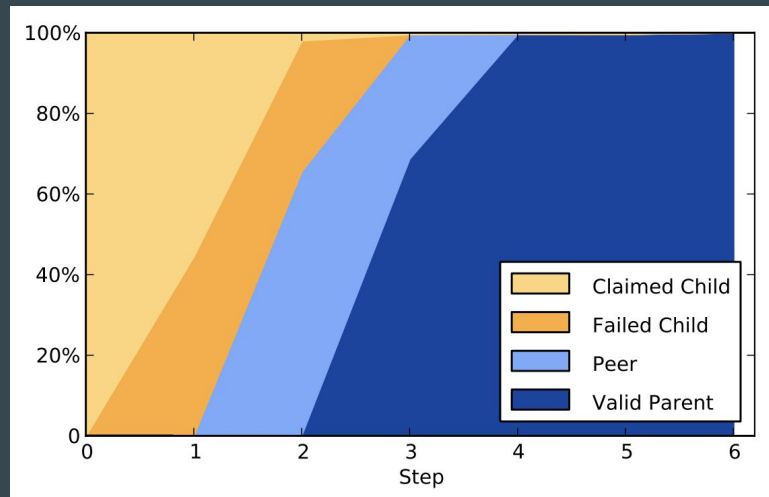
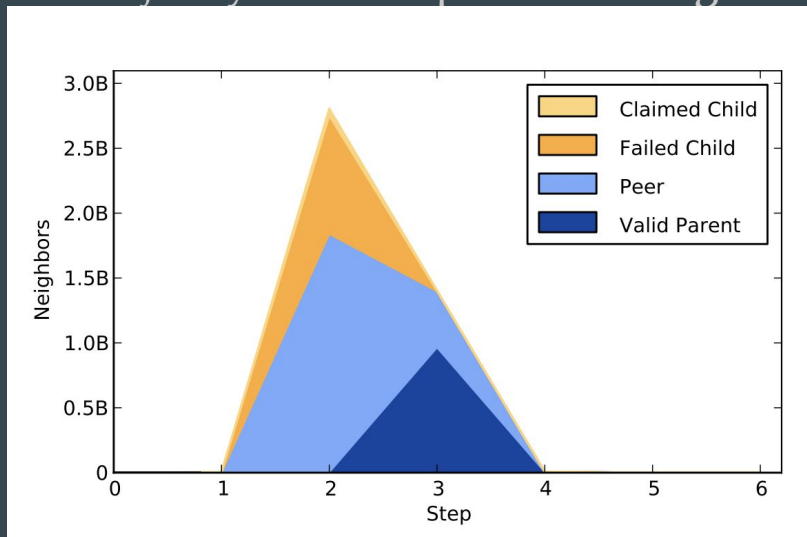


```
function breadth-first-search(vertices, source)
  frontier ← {source}
  next ← {}
  parents ← [-1,-1,...-1]
  while frontier ≠ {} do
    top-down-step(vertices, frontier, next, parents)
    frontier ← next
    next ← {}
  end while
  return tree
```

```
function top-down-step(vertices, frontier, next, parents)
  for v ∈ frontier do
    for n ∈ neighbors[v] do
      if parents[n] = -1 then
        parents[n] ← v
        next ← next ∪ {n}
      end if
    end for
  end for
```

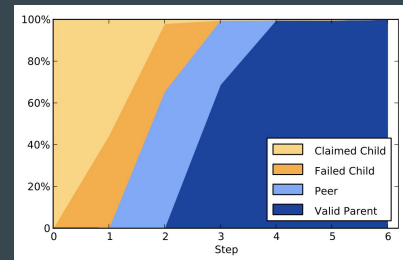
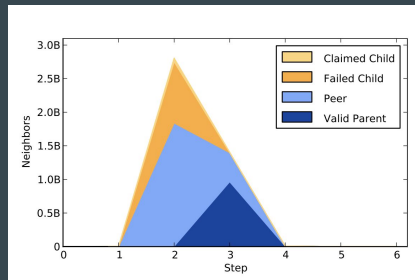
Conventional Top-Down BFS in Social Networks

- Small-world: few steps
 - many vertices visited each step
- Scale-free: frontier size changes rapidly
- Majority of time spent checking if neighbors are visited



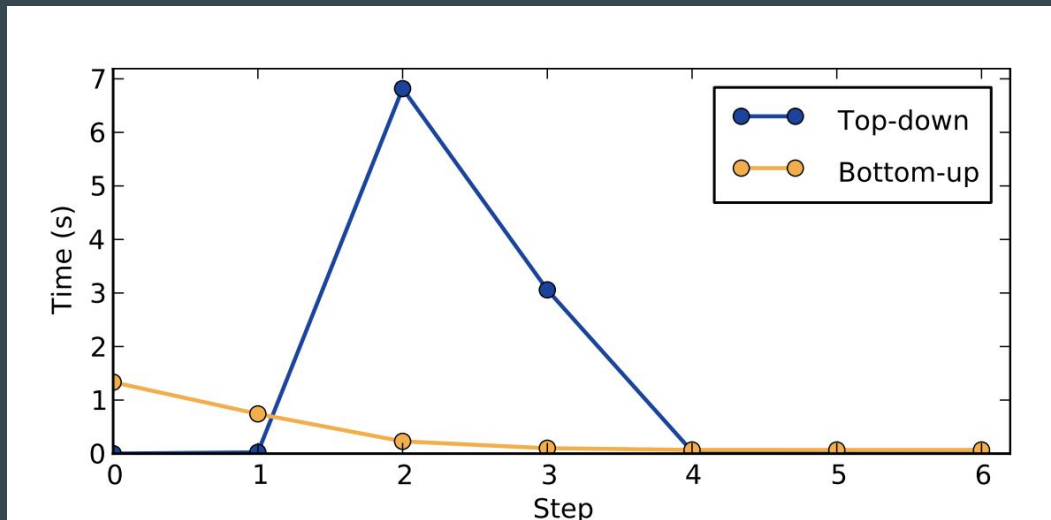
Bottom-Up BFS Approach

- Reduce total edges examined
 - Instead of becoming parent to *all* children, unvisited vertices try to find *any* frontier parent
 - Advantages:
 - don't need to check all neighbors
 - more parallelism (no child contention)
 - Costs:
 - serializes work for one vertex
 - doubles memory for directed graphs



Hybrid Algorithm - Overview

- Small frontier & tail: top-down
- Large frontier: bottom-up
 - switch when frontier largest



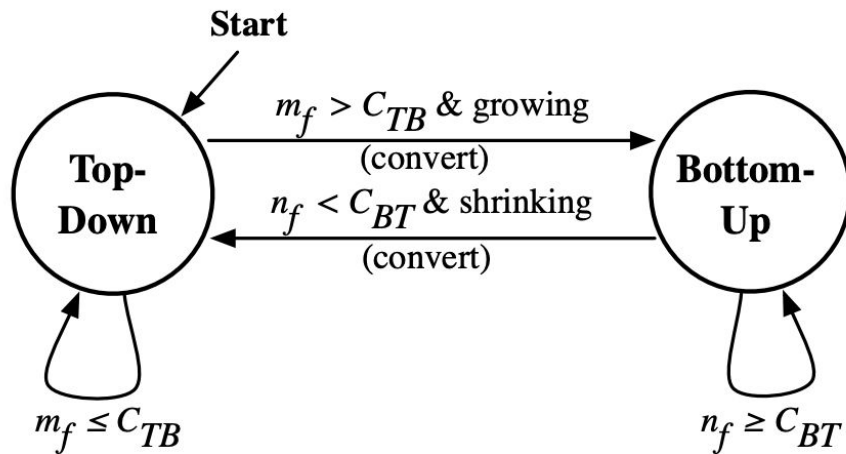
Hybrid Algorithm - Control Flow

- m_f = # edges to check from frontier
- n_f = # vertices in frontier
- m_u = # edges to check from unexplored vertices
- switch:

$$m_f > \frac{m_u}{\alpha} = C_{TB}$$

- switch back:

$$n_f < \frac{n}{\beta} = C_{BT}$$



Hybrid Algorithm - Details

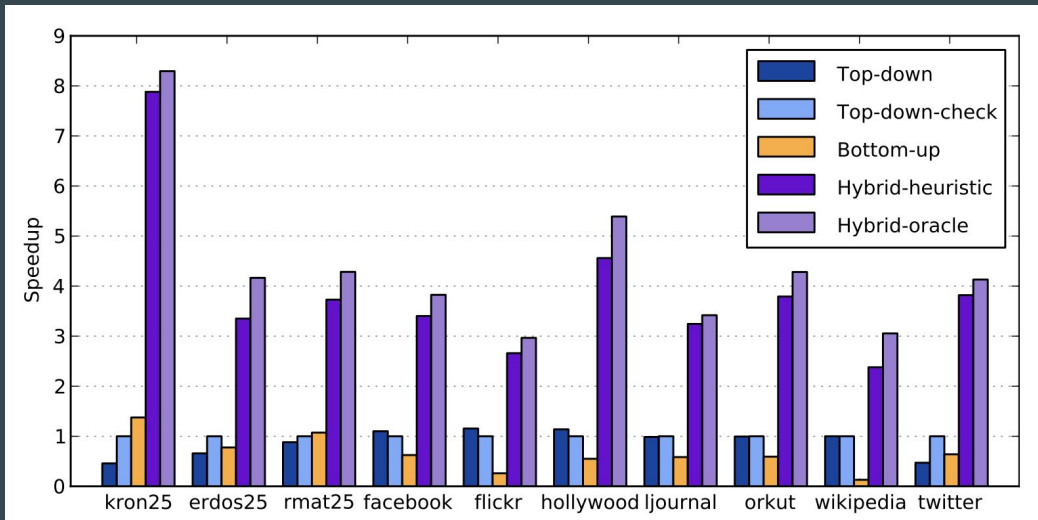
- When switching, need to change representation of frontier from queue to bitmap
- Conversion cost $\ll$ using wrong data structure

Evaluation - Methodology

- Synthetic graphs + real social networks
 - synthetic generated to match Graph500 competition
 - real taken from web crawls
- Three multi-socket server systems
 - 8-core (prior work comparison)
 - 16-core (representative of future computers)
 - 40-core (scalability)
- Implemented with C++ and OpenMP
 - Graph stored in CSR
 - BFS run 64 times from random nonzero vertices, averaged result
- Benchmark: Millions of Edges Traversed per Second (MTEPS)
 - number of edges in the input graph / runtime

Evaluation - Comparing Algorithms

- Baseline: two top-down implementations
 - *top-down*: parallel queue
 - *top-down-check*: adds bitmap to completed nodes
- Two hybrid implementations:
 - *hybrid-heuristic*: as mentioned
 - *hybrid-oracle*: tries all possible switch points, reports best
- 16-core machine



Evaluation - Explanation of Performance Improvement

- A lot less edges examined
- Sites with long tails less speedup - redundancy

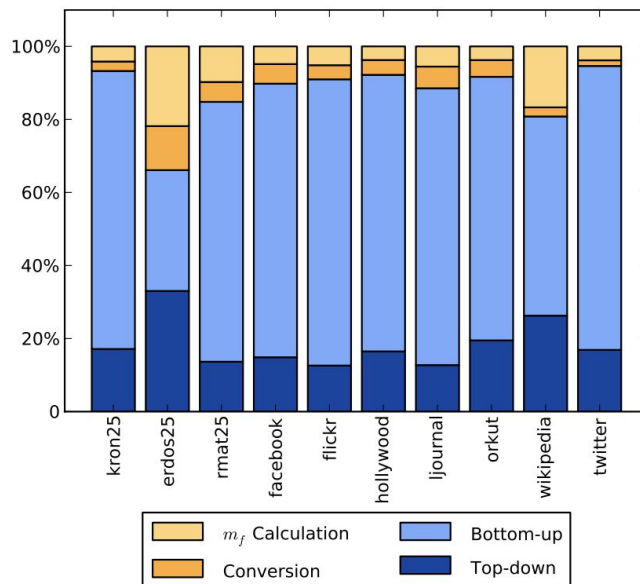
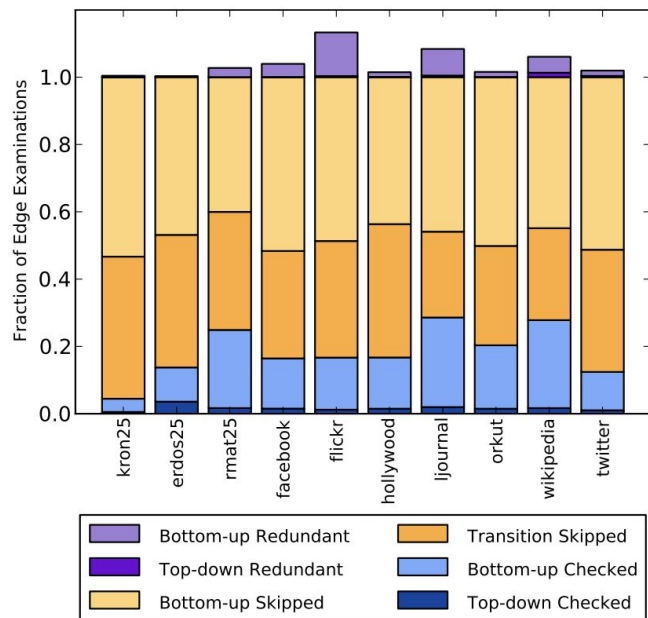
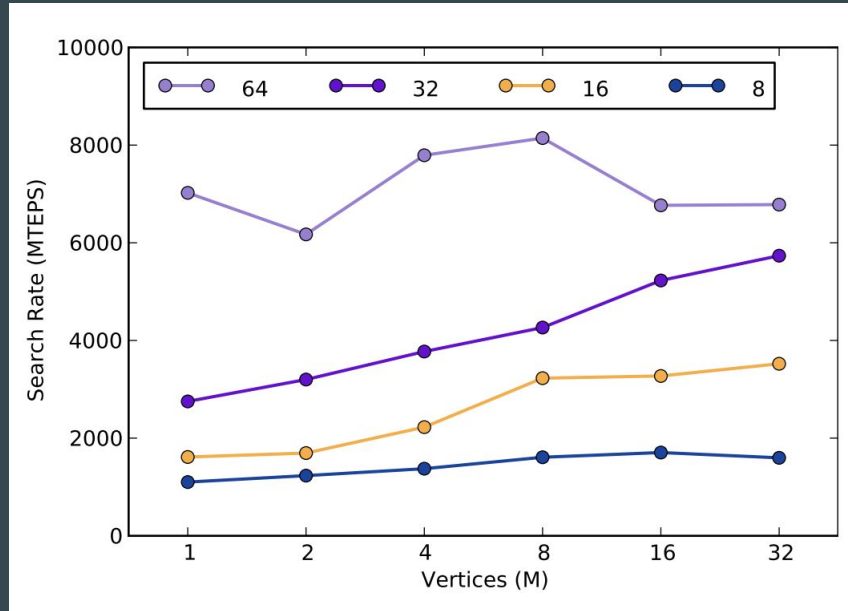
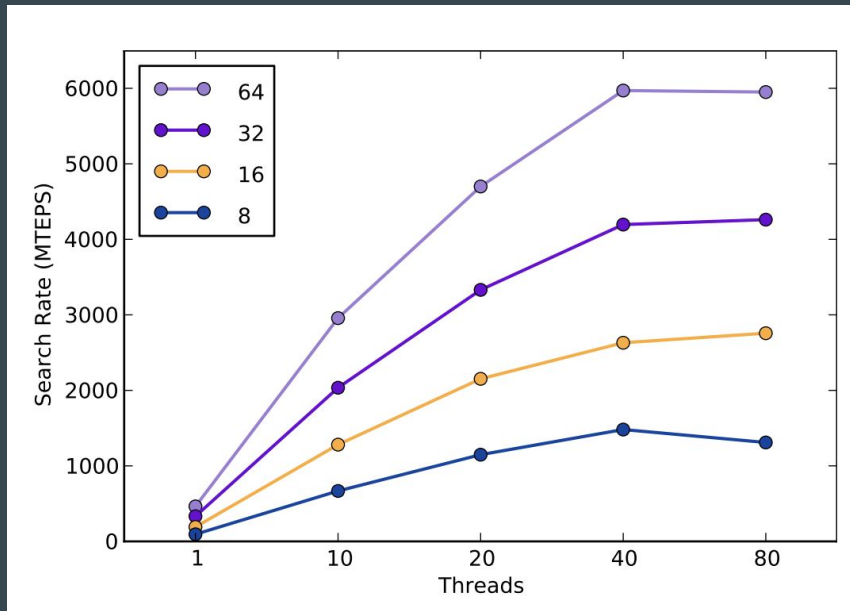


Fig. 12. Breakdown of time spent per search.

Evaluation - Scalability

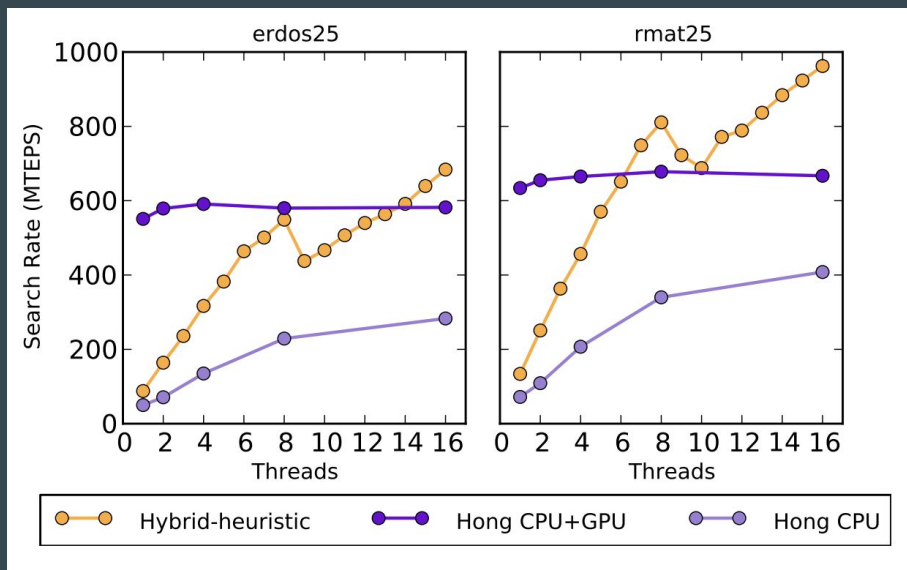
- Parallel speedup with more cores
- Hyperthreading not as effective

- More vertices benefits



Evaluation - Performance Against Other Implementations

- Faster than Hong et al. and Chhugani et al. even with slower hardware



	rmat-8	rmat-32	erdos-8	erdos-32	orkut	facebook
Prior	750	1100	590	1010	2050	920
8-core	1580	4630	850	2250	4690	1360

TABLE III
PERFORMANCE IN MTEPS OF *Hybrid-heuristic* ON THE 8-CORE SYSTEM
COMPARED TO CHHUGANI ET AL. [10]. SYNTHETIC GRAPHS ARE ALL
16M VERTICES, AND THE LAST NUMBER IN THE NAME IS THE DEGREE.

Related Work

Work	Approach	Fewer edge checks?
Bader & Madduri '06 (MTA-2)	Fine-grained hardware threads hide latency; parallelism at vertex and edge level. No caches, so no locality work needed	no
Agarwal et al. SC '10	Locality on commodity quad-socket: thread pinning, socket-local structures, inter-socket message queues, bitmaps to avoid DRAM reads	no
Hong et al. PACT '11	Switches between CPU and GPU at step boundaries via on-line heuristic. "Read-array": frontier fused with output array, depth stored as a byte	no
Merrill et al. PPOPP '12	GPU; prefix sum for frontier expansion offsets → low control divergence, less atomic contention; bitmap filtering	some
Chhugani et al. IPDPS '12	Reordering computation and data layout for spatial locality; dynamic load balancing; analytic performance model	no
This paper	Reverse the search direction when the frontier is dense	yes

My read: strengths

- Simple
- Well-motivated
- Effective
- Graceful degradation - if frontier stays small, bottom-up never used

My read: weaknesses

- Only mean reported - no variance, error bars, etc.
- MTEPS?
- Comparison across more graphs?

Future(?) Direction

- What has happened since
 - Distributed direction optimization by Beamer, Buluç, Asanović, Patterson (IPDPS '13) extended this to 2D-partitioned clusters
 - The technique generalized - Ligra's EDGEMAP, in GraphIt's schedule space, and in GraphBLAS as sparse vs. dense vector operations
 - In the GAP Benchmark Suite, the direction-optimizing BFS is the reference implementation
- Open tasks
 - Different implementations?
 - Test with more modern hardware - more cores, memory, etc.

Questions for discussion

- MTEPS as a metric?
- Ways to switch between top-down and bottom-up without manually tuned α , β ?
- Extra memory trade-off for directed graphs?
- Generalization of direction optimization?
- Other graph properties?

Thank You For Listening

...

AI Usage

- I used AI to help generate & prepare for the discussion questions