GAIL
The Graph Algorithm Iron Law

Scott Beamer, Krste Asanović, David Patterson

ASPIRE

GAP

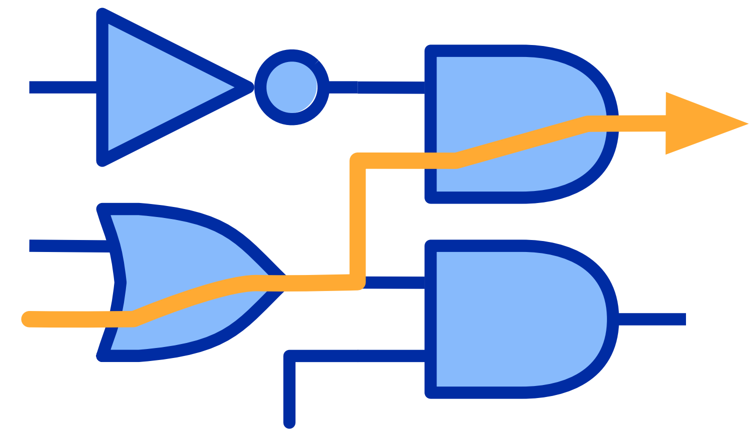
Berkeley
ELECTRICAL ENGINEERING
& COMPUTER SCIENCES

gap.cs.berkeley.edu

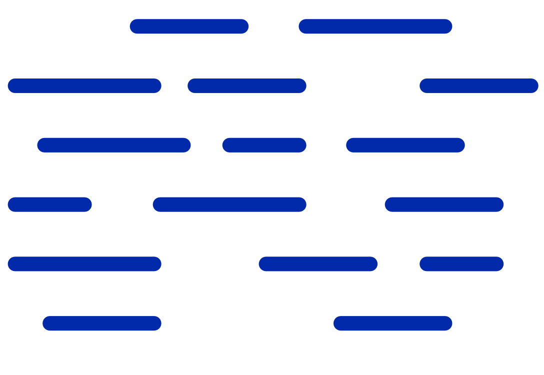
Graph Applications

Social Network Analysis

Recommendations

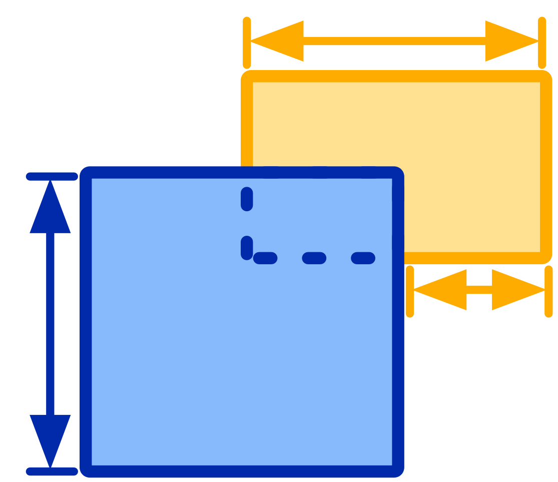


CAD



Bioinformatics

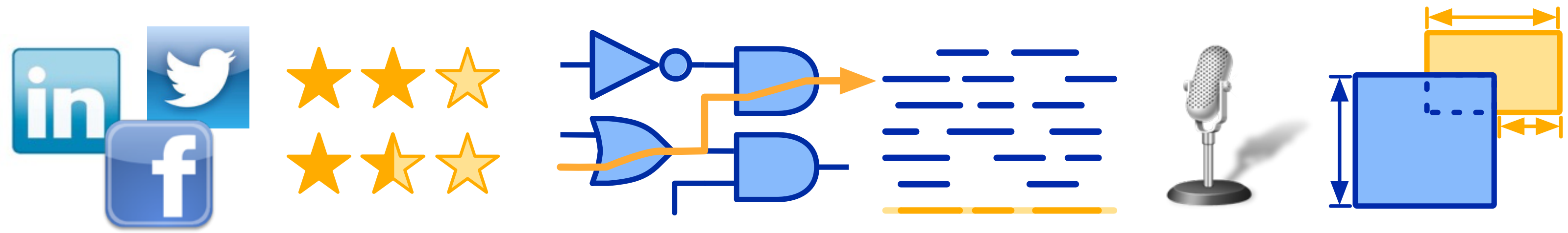
Speech Recognition



Webpage Layout

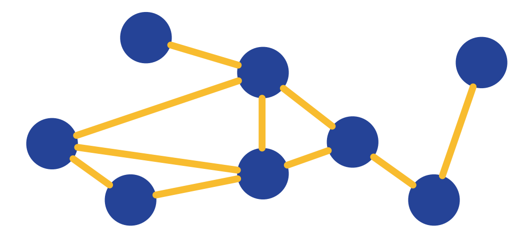
Research Ongoing At All Levels

Applications



GAIL

Algorithms



Implementation

Platform

Need More Informative Metrics

ASPIRE UC Berkeley

- Time
  - \+ is often the the most important
  - \- requires other parameters matched
- Traversed edges per second (TEPS)
  - \+ a rate, so can compare different inputs
  - \- confusion about what counts as a TE

Need More Informative Metrics

- Time
  - \+ is often the the most important
  - \+ a rate, so can compare different inputs
  - \- confusion about what counts as a TE

Time & TEPS only quantify which is fastest, no insight into why

Graph Performance Factors

① Algorithms - how much work to do

② Cache utility - how much data to move

③ Memory bandwidth - how fast data moves

- For measurements: [*Beamer, IISWC, 2015*]

Graph Algorithm Iron Law (GAIL)

$$\frac{\text{time}}{\text{kernel}} = \frac{\text{edges}}{\text{kernel}} \times \frac{\text{mem. req.}}{\text{edge}} \times \frac{\text{time}}{\text{mem. req.}}$$

Graph Algorithm Iron Law (GAIL)

ASPIRE
UC Berkeley

annotate code to count *edges traversed*

$$\frac{\text{time}}{\text{kernel}} = \frac{\text{edges}}{\text{kernel}} \times \frac{\text{mem. req.}}{\text{edge}} \times \frac{\text{time}}{\text{mem. req.}}$$

use performance counters to access total # of *memory requests*

Graph Algorithm Iron Law (GAIL)

ASPIRE UC Berkeley

Role: algorithm designer

$$\frac{\text{time}}{\text{kernel}} = \boxed{\frac{\text{edges}}{\text{kernel}}} \times \frac{\text{mem. req.}}{\text{edge}} \times \frac{\text{time}}{\text{mem. req.}}$$

Metric: algorithmic intensity

Graph Algorithm Iron Law (GAIL)

ASPIRE UC Berkeley

Role: implementor

$$\frac{\text{time}}{\text{kernel}} = \frac{\text{edges}}{\text{kernel}} \times \boxed{\frac{\text{mem. req.}}{\text{edge}}} \times \frac{\text{time}}{\text{mem. req.}}$$

Metric: cache utility

Graph Algorithm Iron Law (GAIL)

ASPIRE UC Berkeley

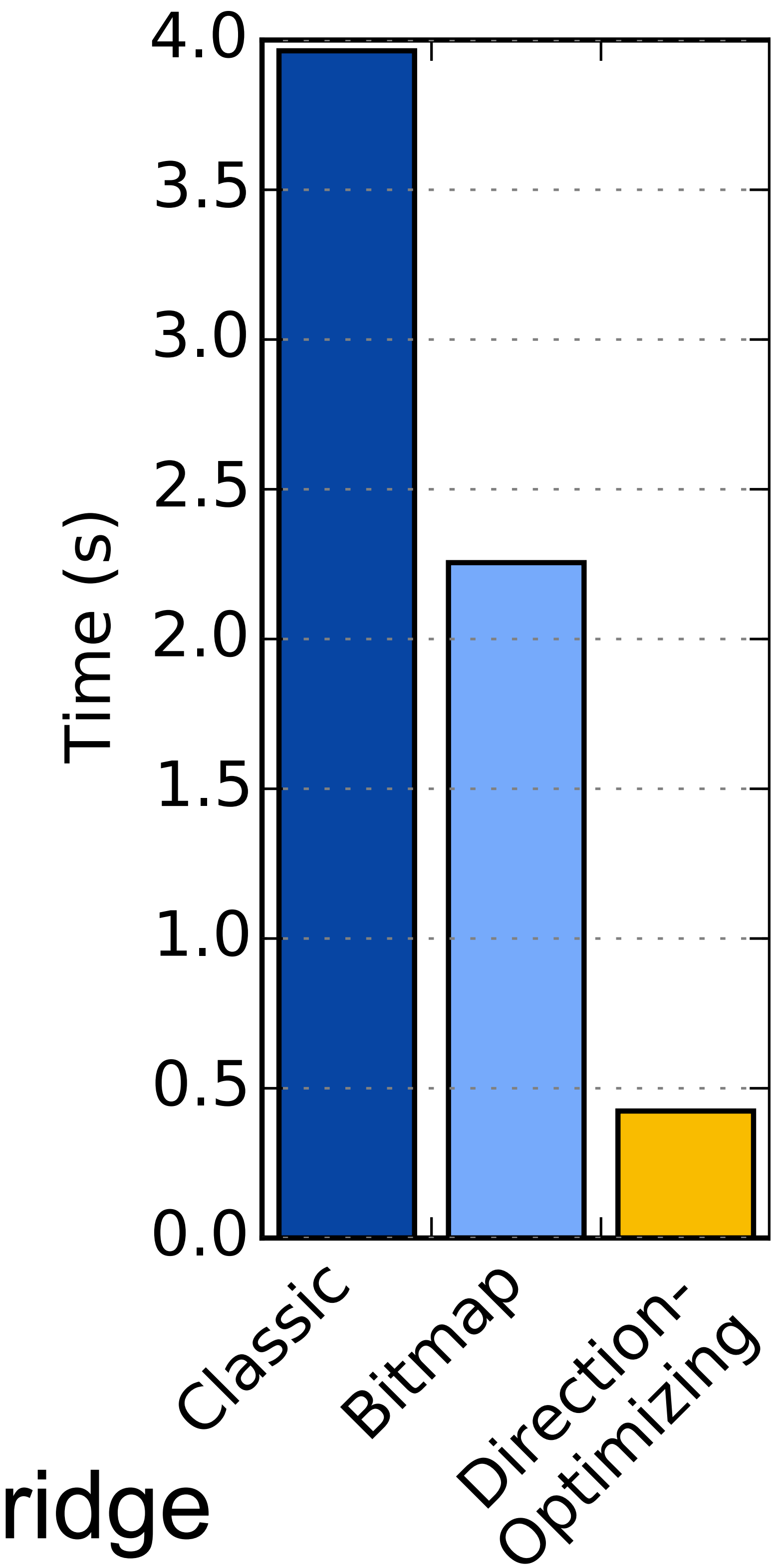
Role: system designer

$$\frac{\text{time}}{\text{kernel}} = \frac{\text{edges}}{\text{kernel}} \times \frac{\text{mem. req.}}{\text{edge}} \times \boxed{\frac{\text{time}}{\text{mem. req.}}}$$

Metric: DRAM BW utilization

Comparing BFS Implementations

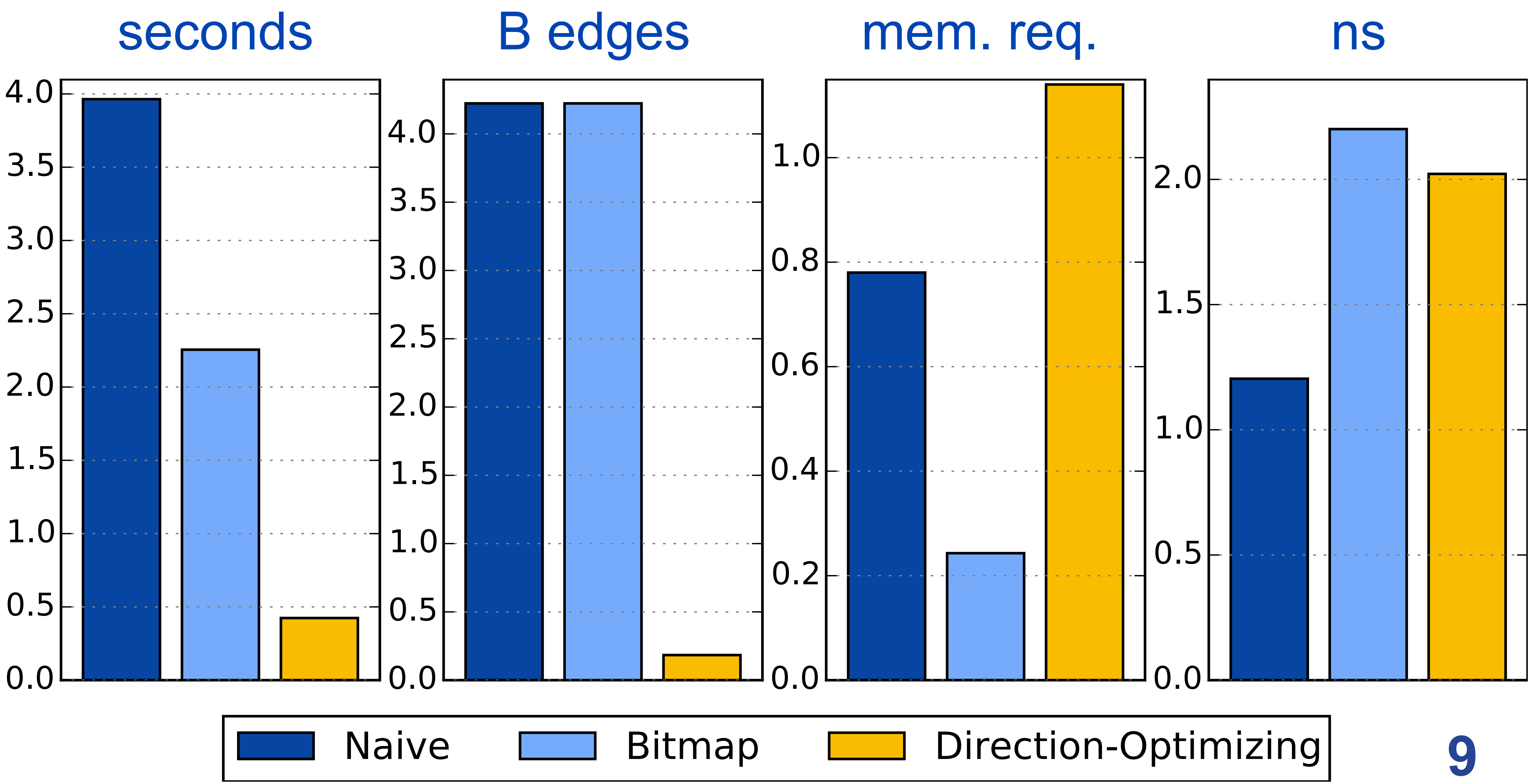
- 3 BFS Approaches
  - *Naive* - classic top-down
  - *Bitmap* - uses bitmaps to reduce communication
  - *Direction-optimizing* - algorithmically does less
- Time doesn't explain speedup



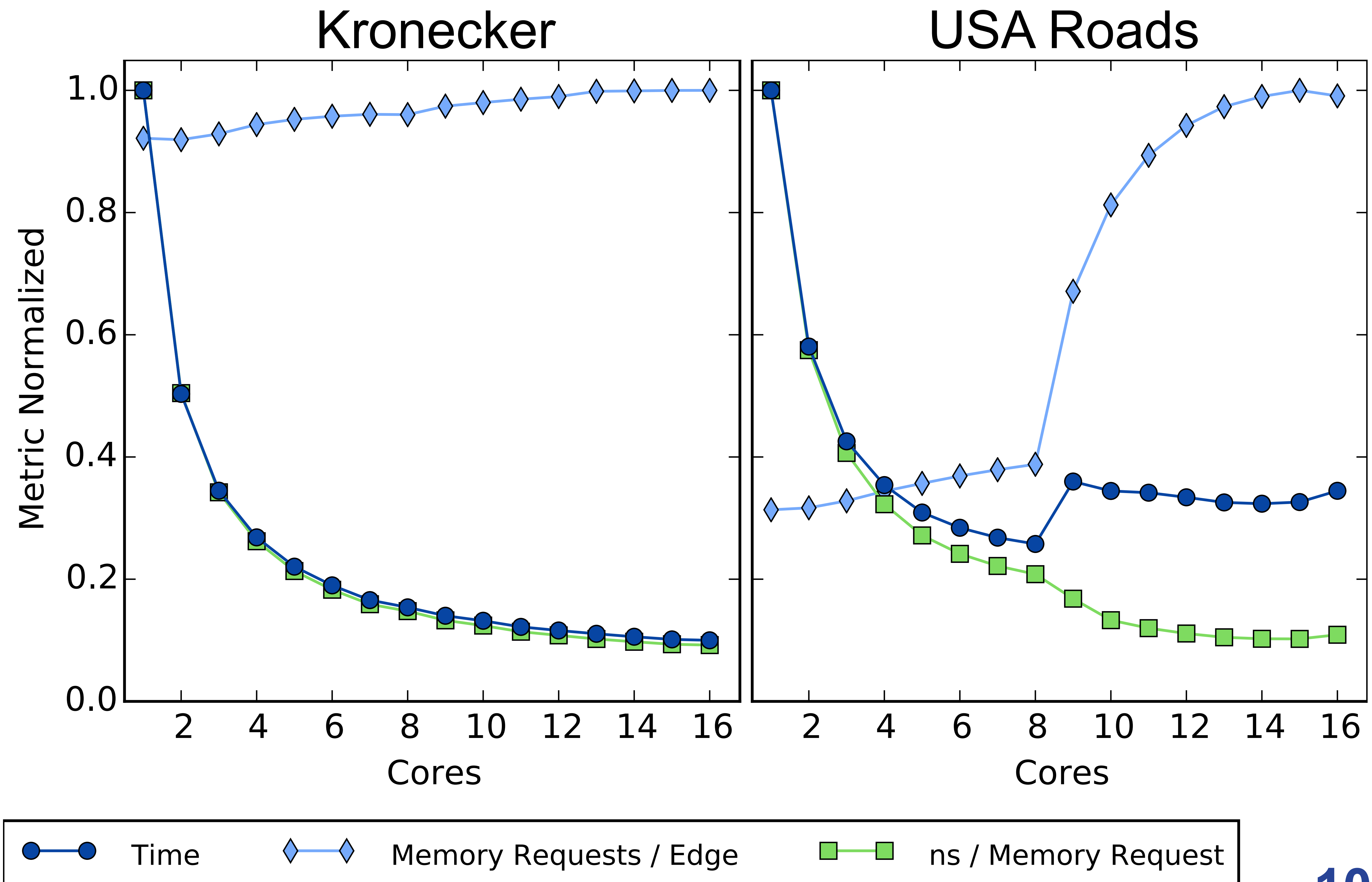
Kronecker SCALE=27, 32 threads, Ivy Bridge

BFS Analyzed by GAIL

$$\frac{\text{time}}{\text{kernel}} = \frac{\text{edges}}{\text{kernel}} \times \frac{\text{mem. req.}}{\text{edge}} \times \frac{\text{time}}{\text{mem. req.}}$$

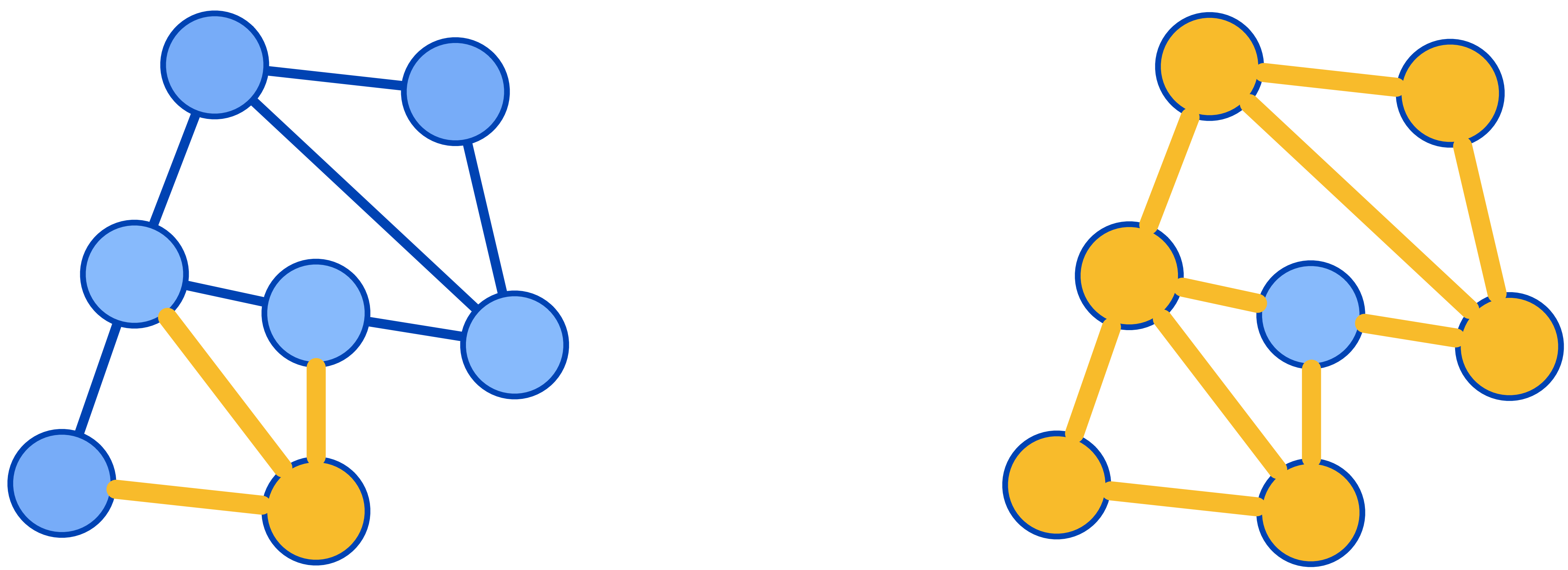


BFS Strong Scaling Analyzed by GAIL



Delta-Stepping Analyzed by GAIL

- Single-source shortest paths algorithm

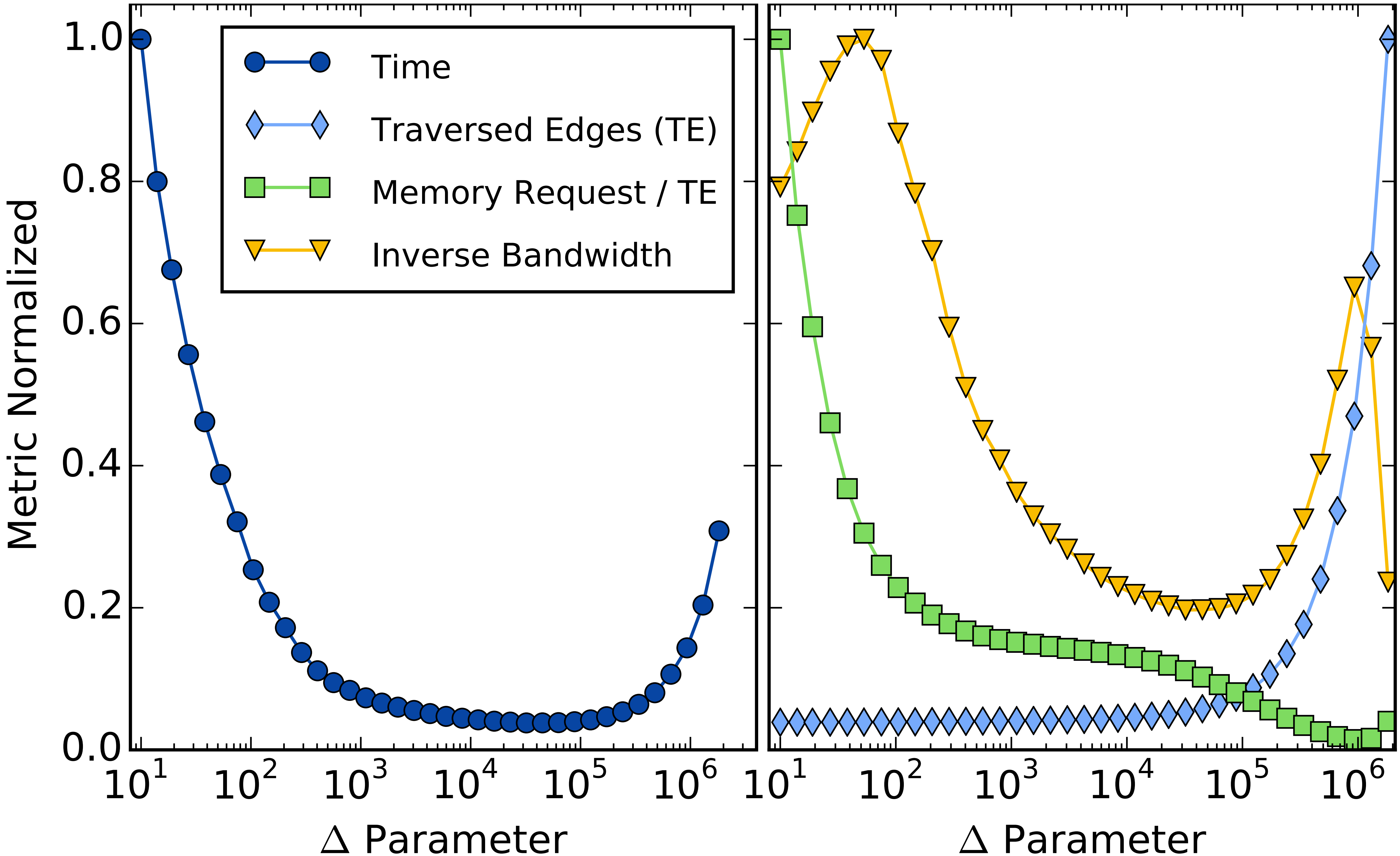


1 — Δ Parameter — ∞

~Dijkstra — ~Bellman-Ford

Work Efficiency ⟵ tradeoff ⟶ Parallelism

Delta-Stepping Analyzed by GAIL



USA roads, 8 threads, Ivy Bridge

GAP Benchmark Suite

GAP

- Benchmark Specifications (technical report)
  - standardize input graphs and rules
  - allows other implementations to compare
- Portable, high-quality baseline code
  - Only requirement is C++11 & OpenMP
  - Built in testing to verify results

gap.cs.berkeley.edu

Conclusion

$$\frac{\text{time}}{\text{kernel}} = \frac{\text{edges}}{\text{kernel}} \times \frac{\text{mem. req.}}{\text{edge}} \times \frac{\text{time}}{\text{mem. req.}}$$

gap.cs.berkeley.edu

Conclusion

ASPIRE UC Berkeley

$$\frac{\text{time}}{\text{kernel}} = \frac{\text{edges}}{\text{kernel}} \times \frac{\text{mem. req.}}{\text{edge}} \times \frac{\text{time}}{\text{mem. req.}}$$

- GAIL concisely breaks down performance
  - useful as a starting point for **introspection**
  - useful as simple model to weigh **tradeoffs**

gap.cs.berkeley.edu

Acknowledgements

Research partially funded by DARPA Award Number HR0011-12-2-0016, the Center for Future Architecture Research, a member of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA, and ASPIRE Lab industrial sponsors and affiliates Intel, Google, Huawei, Nokia, NVIDIA, Oracle, and Samsung. Any opinions, findings, conclusions, or recommendations in this paper are solely those of the authors and does not necessarily reflect the position or the policy of the sponsors.

Bonus Slides

What Do GAIL Results Represent?

- GAIL results are for a particular execution
  - fixed: input, platform, implementation
  - changing any of above will change results
- Focused on single-server shared-memory
- GAIL requirements
  - measure: runtime, traversed edges, memory requests
  - algorithm has notion of “traversing” edge

Why Not Complexity Analysis?

- Formal complexity analysis is helpful, but…
  - Many algorithms' runtime is input graph topology-dependent, but often difficult to model real-world graphs
  - Hides many improvements to platform or implementation optimizations
  - Can be overly pessimistic. Many algorithms with a slower worst-case performance much faster in practice

What About Other Platforms?

- GAIL is for single-server shared memory
- For other platforms, replace memory requests with equivalent bottleneck metric
  - Clusters: packets or bytes transmitted
  - Flash/HD: blocks read from storage
  - Cache-less (XMT): memory requests OK

Iron Law Reapplied

For CPUs:

$$\frac{\text{time}}{\text{program}} = \frac{\text{insts.}}{\text{program}} \times \frac{\text{cycles}}{\text{inst.}} \times \frac{\text{time}}{\text{cycle}}$$

For Graph Algorithms:

$$\frac{\text{time}}{\text{kernel}} = \frac{\text{edges}}{\text{kernel}} \times \frac{\text{mem. req.}}{\text{edge}} \times \frac{\text{time}}{\text{mem. req.}}$$

Graph Algorithm Iron Law (GAIL)

ASPIRE UC Berkeley

$$\frac{\text{time}}{\text{edge}} = \frac{1}{\text{TEPS}}$$

$$\frac{\text{time}}{\text{kernel}} = \frac{\text{edges}}{\text{kernel}} \times \frac{\text{mem. req.}}{\text{edge}} \times \frac{\text{time}}{\text{mem. req.}}$$

$$\frac{\text{mem. req.}}{\text{kernel}} = \frac{\text{data transferred}}{\text{cache line size}}$$

Evaluation Setup

Graph	# Vertices	# Edges	Degree	Diameter	Degree Dist.
Roads of USA	23.9M	58.3M	2.4	High	const
Web Crawl of .sk Domain	50.6M	1949.4M	38.5	Medium	power
Kronecker Synthetic Graph	128.0M	2048.0M	16.0	Low	power

- Target Platform
  - Dual-socket Intel Ivy Bridge 3.3 GHz
  - Socket: 8 cores with 25MB L3 cache
  - DRAM: 128 GB DDR3-1600