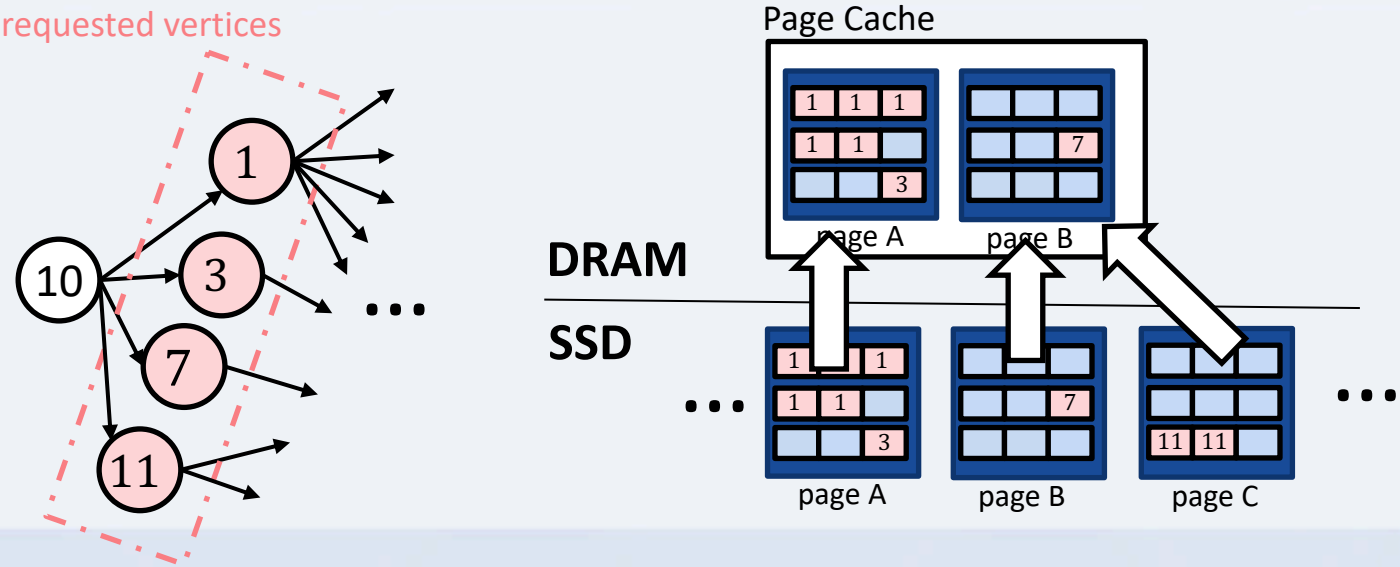
Pre-Select Static Caching and Neighborhood Ordering for BFS-like Algorithms on Disk-based Graph Engines

Eunjae Lee, Junghyun Kim, Keunhak Lim, Sam H. Noh, Jiwon Seo

UNIST, Hanyang University

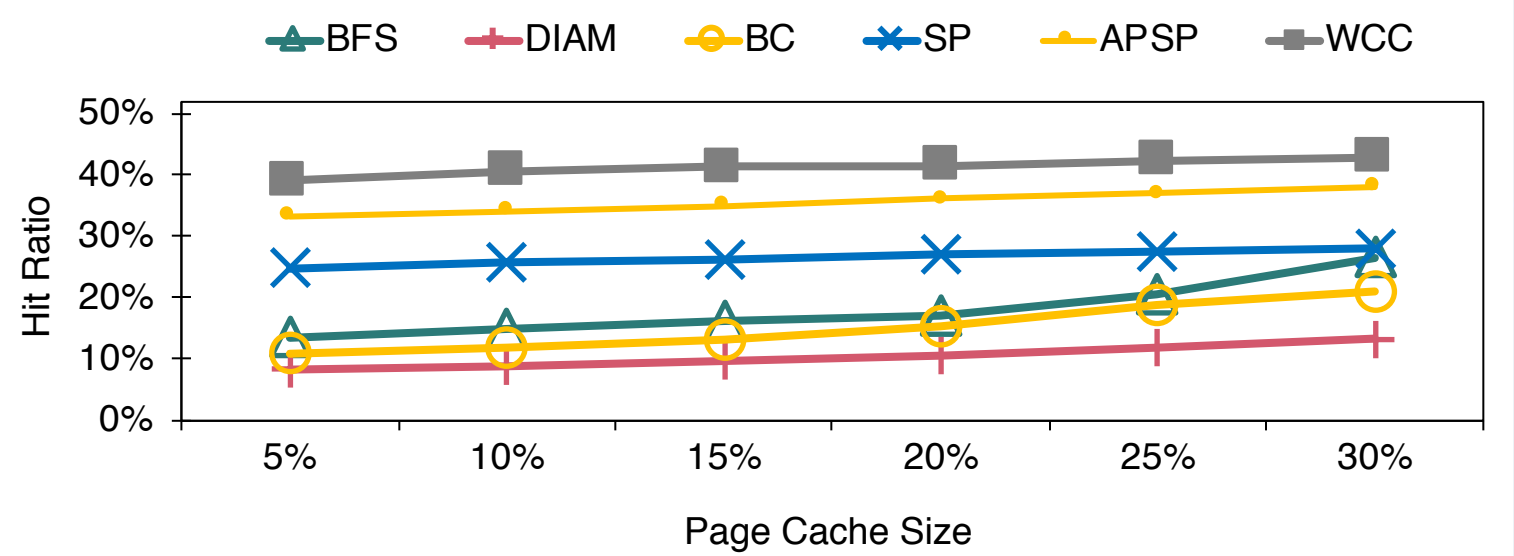
BFS-like Algorithms in Disk-based Graph Engine

- ***BFS-like Algorithms*: recursive graph traversal**
  - Eg. *BFS*, *Shortest Paths*, *Betweenness Centrality*, ...
- **Disk-based Graph Engine**
  - Graph (edge lists) is stored on disk
  - When vertices are visited, their edge lists are loaded to page cache



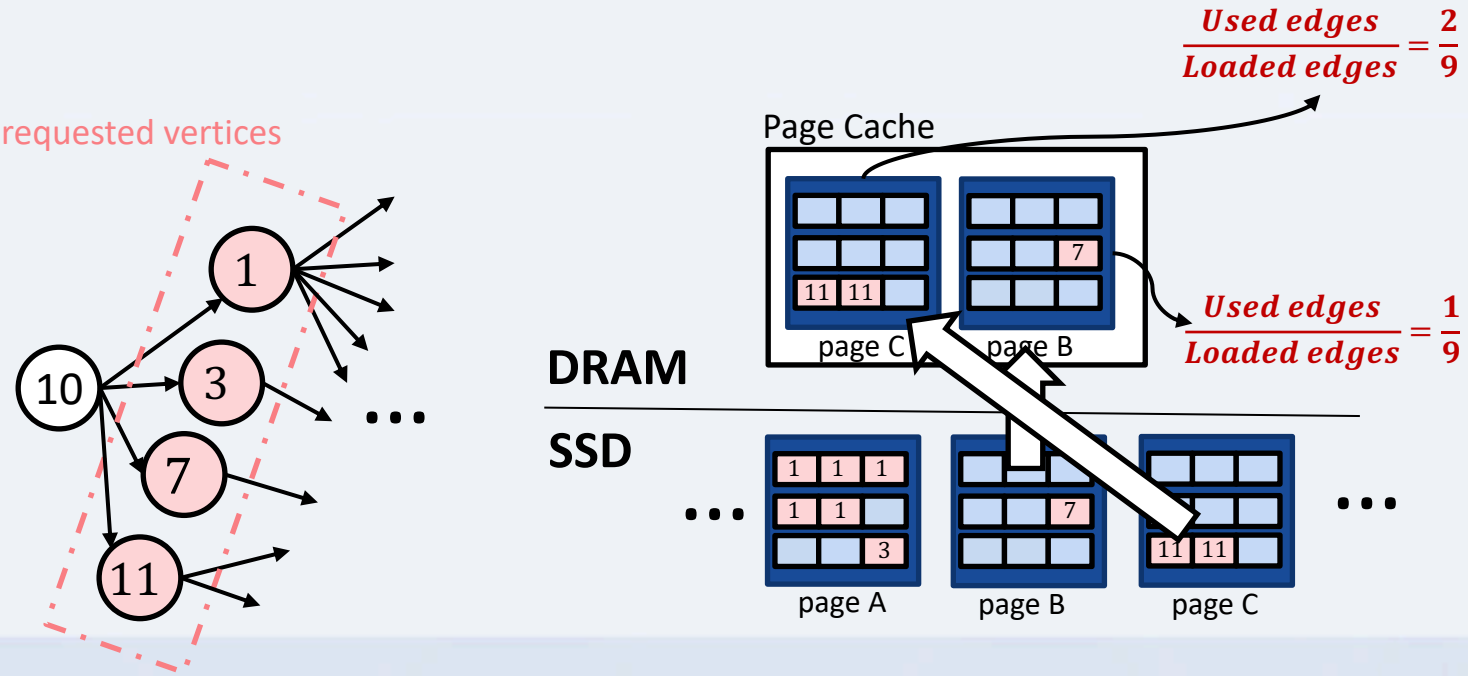
BFS-like Algorithms in Disk-based Graph Engine

- **Page cache is inefficient for BFS-like algorithm**
  1. Increasing size of page cache **does not** help performance



BFS-like Algorithms in Disk-based Graph Engine

- **Page cache is inefficient for BFS-like algorithm**
  1. Increasing size of page cache **does not** help performance
  2. Utilization of page cache is **low**

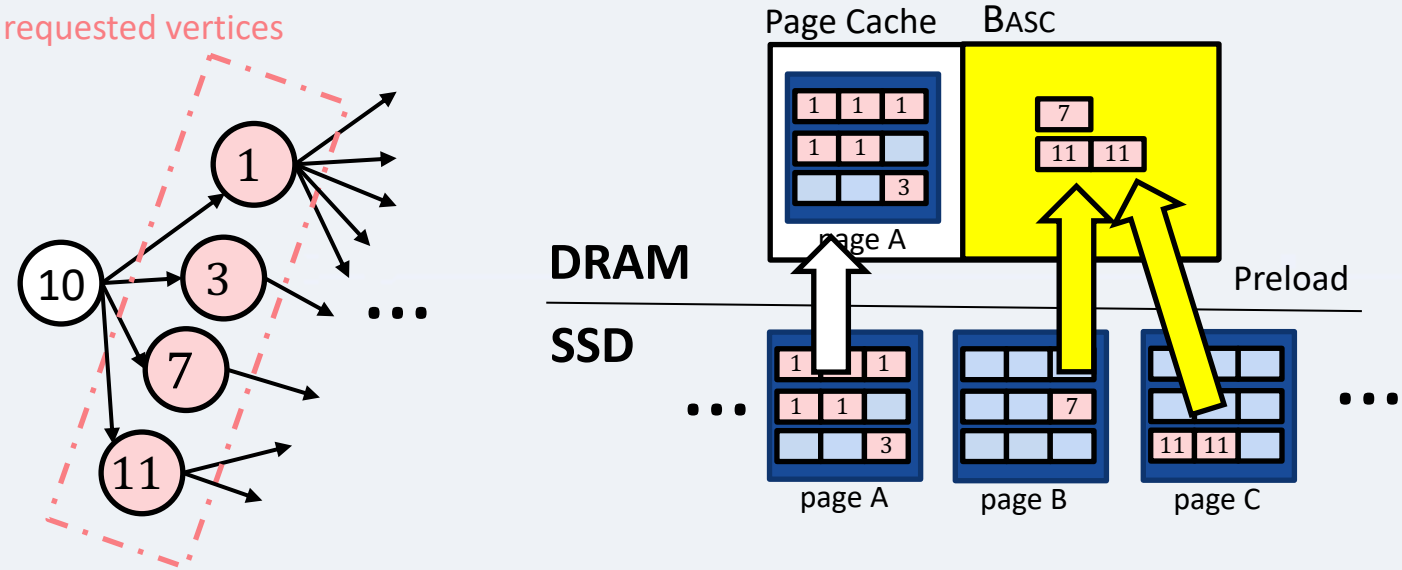


Our Optimization

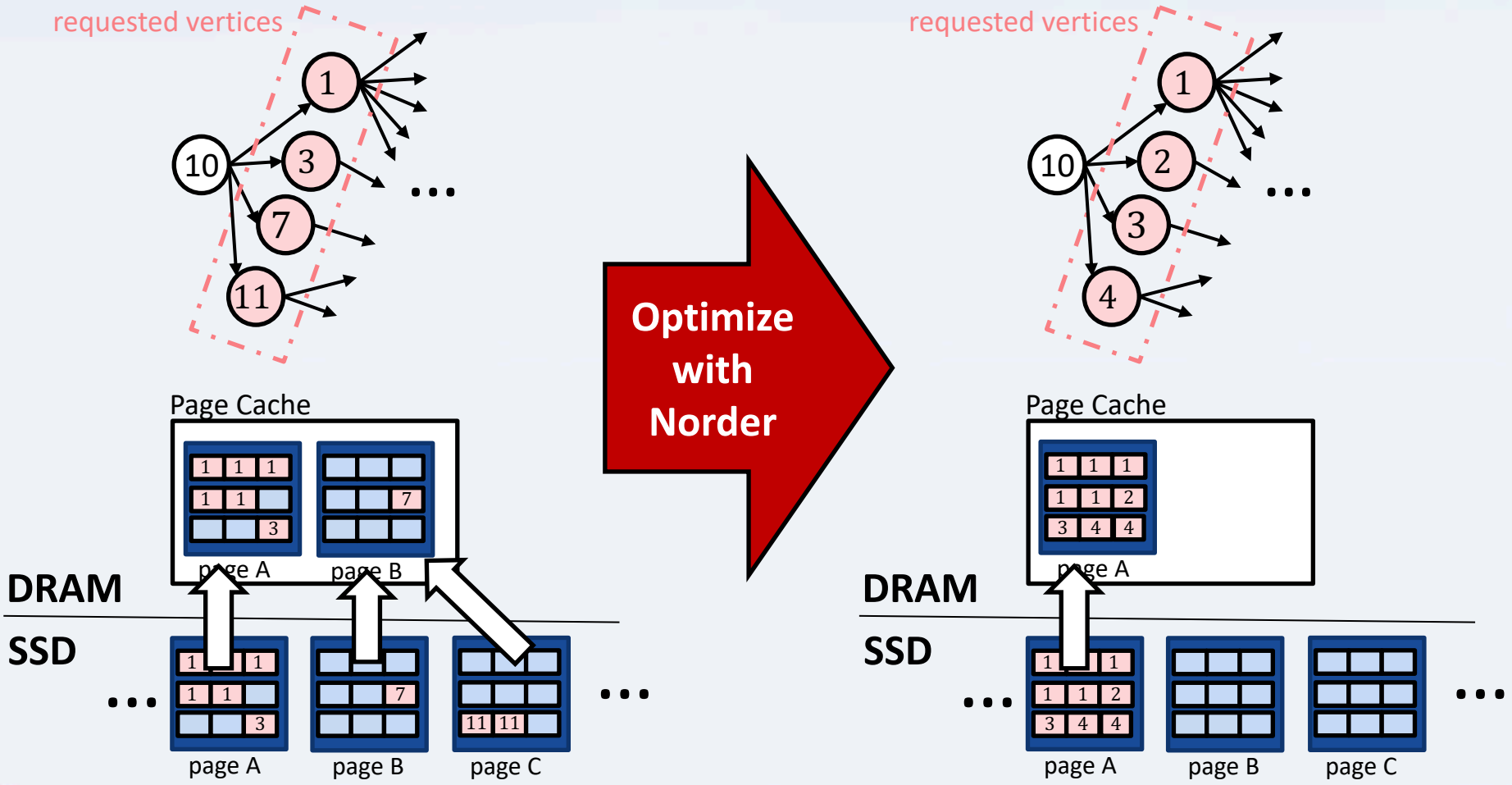
- **BFS-Aware Static Cache (BASC)**
  - Keep separate cache for selected edge lists
  - Pre-loaded: edges pre-selected through pre-analysis
  - Static: contents of cache does not change

- **Neighborhood Ordering (Norder)**
  - Graph ordering optimization for better memory utilization

1. BASC: BFS-Aware Static Cache

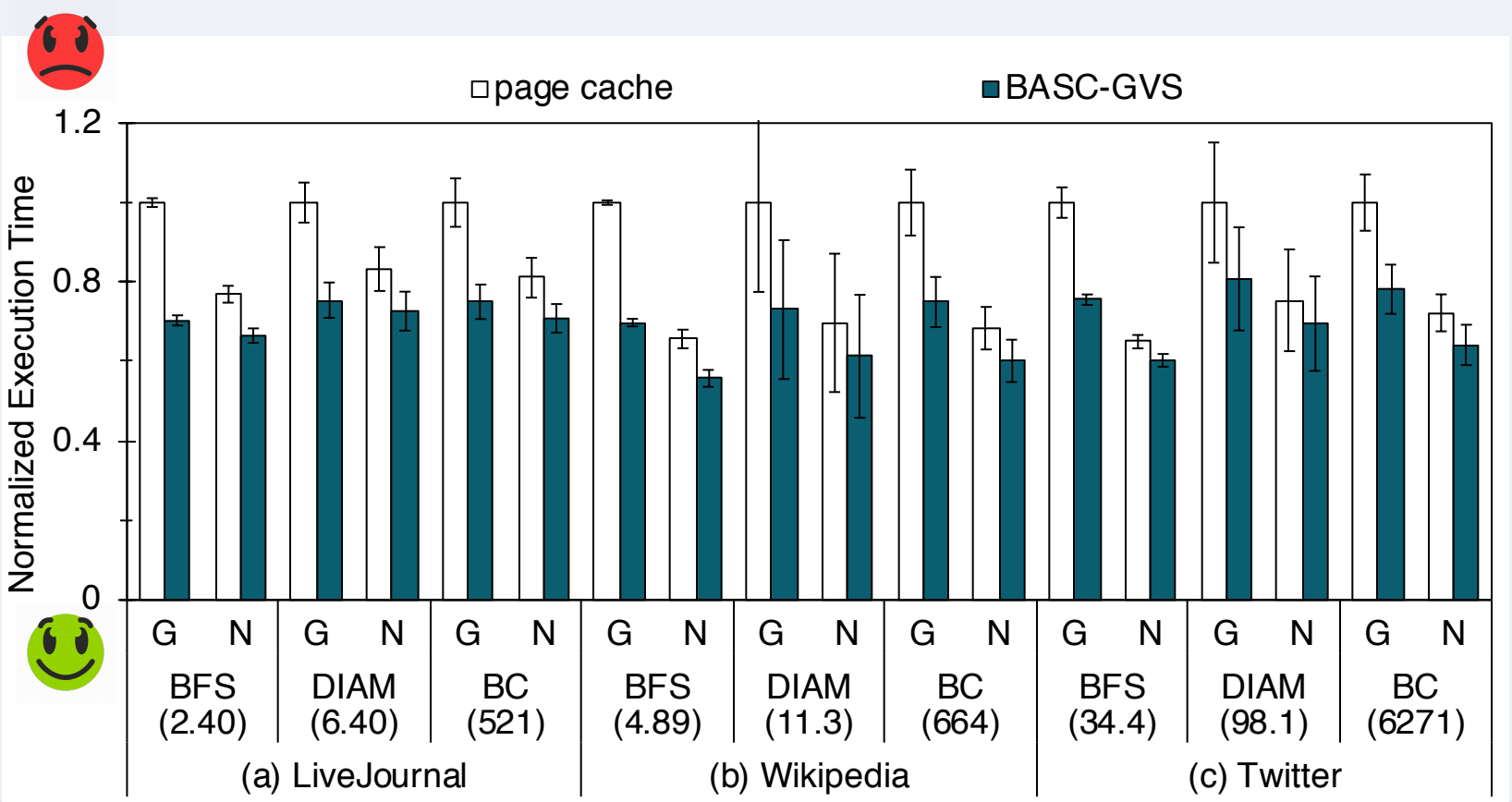


2. Norder: Neighborhood Ordering



Performance Improvement

- **BASC + Norder: 54% faster than Gorder* + Page Cache**
  - Tested with 7 BFS-like algorithms & 5 data sets



- Total cache size is 25% of graph file size
- Gorder is state-of-the-art ordering method [H. Wei, SIGMOD'16]

Presentation

- **Time**
  - Thursday, July 11, 2019
  - Track 2: Graph Processing Frameworks
  - 11:15 AM – 12:35 PM, 4th presentation

- **Authors attending ATC ’19**

Eunjae Lee

Jiwon Seo

Sam H. Noh