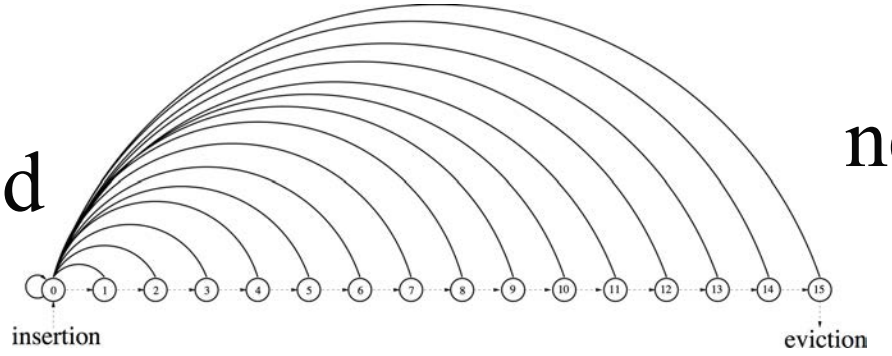
Insertion and Promotion for Tree-Based PseudoLRU Last-Level Caches

- Daniel A. Jiménez, Texas A&M University
- LRU keeps blocks in a recency stack
  - *n*-way cache, 0 is MRU, *n*-1 is LRU
- When a block is inserted or promoted (used) it goes to the MRU position
  - Not always the best choice
- Instead, let's use the blocks' former position to indicate its new position

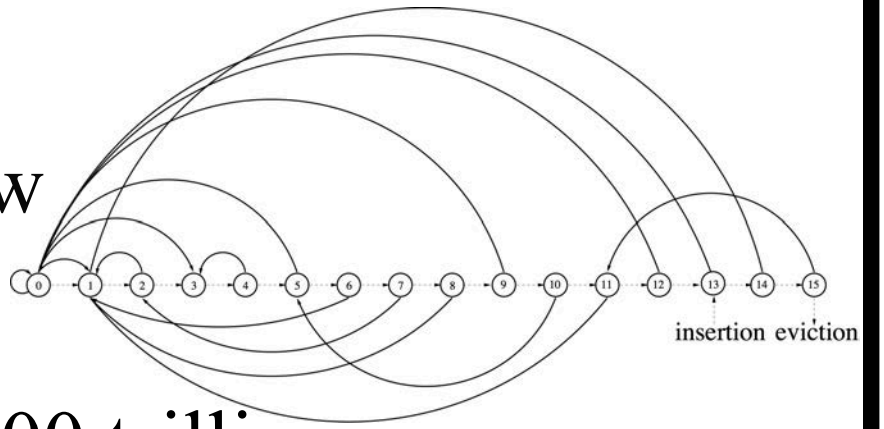
Searching a Large Design Space

- We want to develop a new transition graph

old



new



- For 16-way, there are > 300 trillion possibilities
- So we use a genetic algorithm to search them
  - Fitness function is estimate of speedup

PseudoLRU instead of LRU

- This idea works just as well for tree-based PseudoLRU
- Use set-dueling to dynamically choose between policies
- Replacement policy consumes < 1 bit per block
- Performance comparable to state-of-the-art
  - 5.6% speedup over LRU on SPEC CPU 2006
  - 15.6% on a memory-intensive subset