MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS

Gabriel Moruz
University of Frankfurt

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Danmarks Grundforskningsfond
Danish National Research Foundation

OnlineMIN: A Fast Strongly Competitive Paging Algorithm

Paging and Competitive Analysis

Paging

- Setup: a cache of size k and a memory of infinite size
- Process pages sequentially online (no information about future)
- Current page:
  - Cache hit – page is in cache: move to next page
  - Cache miss – page is not in cache: bring it in cache
    - Cache is full: evict some page to make room
- Objective: minimize #misses

Competitive analysis

- Compare online algorithm against optimal cost OPT
- An algorithm A has competitive ratio c if

$$cost(A) \le c \times cost(OPT)$$

Results

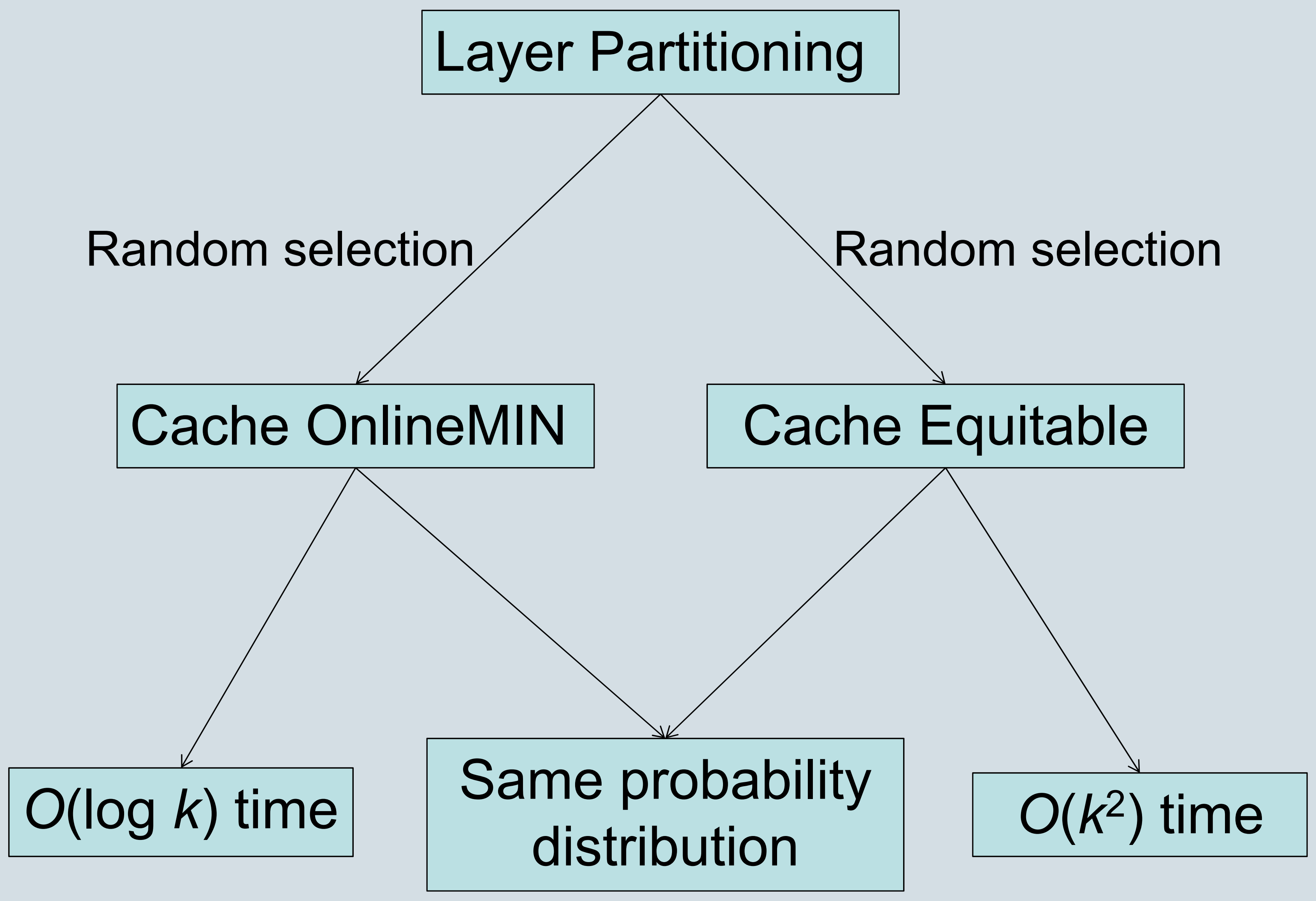
Previous work

Algorithm	Comp. ratio	Space	Time per page
LRU, FIFO	k	$O(k)$	$O(1)$
Mark	$2H_k$	$O(k)$	$O(1)$
Partition	H_k	$O(n)$	$O(n)$
Equitable	H_k	$O(k^2 \log k)$	$O(k^2)$
Equitable2	H_k	$O(k)$	$O(k^2)$

OnlineMIN

H_k-competitive, $O(k)$ space, $O(\log k)$ *time per page*

Roadmap



Layer Partitioning

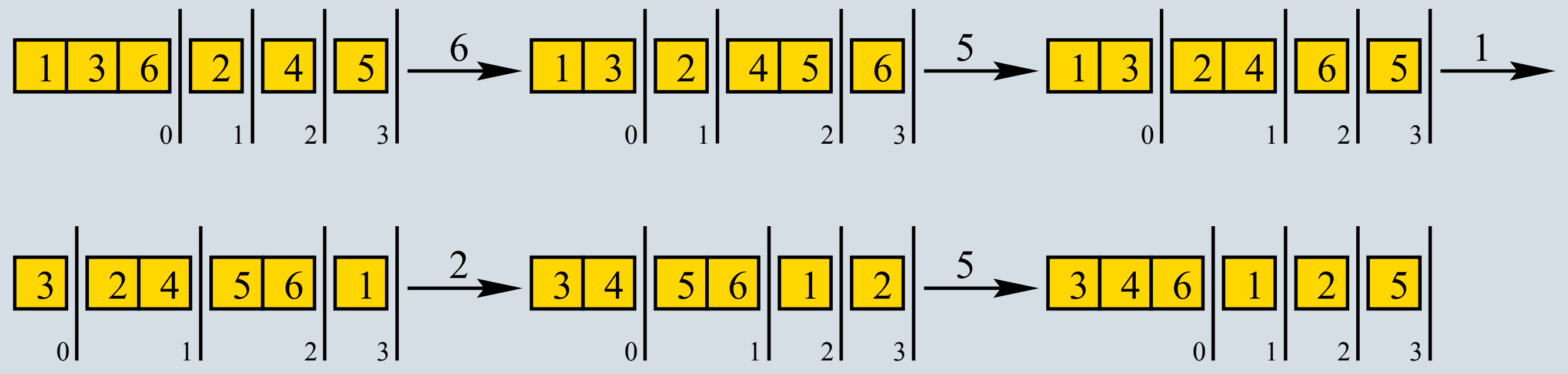
Layers

- Intuition: keep track of OPT's cache
- Split all pages in k+1 layers $L_0, \ldots, L_k$
- At most i pages in first i layers in OPT's cache

Layer update upon request to page p:

- P in L_0:
  - $L_0 = L_0 - \{p\}$, $L_{k-1} = L_{k-1} + L_k$, $L_k = \{p\}$
- P in L_i ($i > 0$):
  - $L_{i-1} = L_{i-1} + L_i - \{p\}$, $L_j = L_{j+1}$ (for $j \ge i$), $L_k = \{p\}$

Example (k = 3)

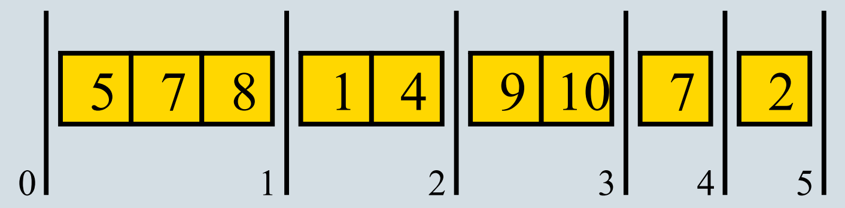


Selection Process

Selection process

- Assume pages have random priorities
- Build sets $C_0, \ldots, C_k$ as follows
  - $C_0 = \{\}$
  - C_i has the i pages in $C_{i-1} + L_i$ having largest priorities
- The cache of the algorithm is always C_k

Example (k = 5)



$C_1 = \{8\}$, $C_2 = \{4,8\}$, $C_3 = \{8,9,10\}$, $C_4 = \{7,8,9,10\}$, $C_5 = \{2,7,8,9,10\}$

Same distribution as Equitable2, and thus H_k-competitive!

Implementation

OnlineMIN

- Upon processing page p:
- Update cache if cache miss:
  - If p in L_0, evict page in cache having smallest priority
  - If p in L_i ($i > 0$)
    - Find smallest $j > i$ s.t. first j layers have j pages in cache
    - Evict the page in the first j layers having smallest priority
- Update layers as previously described

Analysis

$O(k)$ space per Equitable2, $O(\log k)$ time per smart data structures

References

[1] Gerth Stølting Brodal, Gabriel Moruz, and Andrei Negoescu. *OnlineMin: A Fast Strongly Competitive Randomized Paging Algorithm*. Theory of Computing Systems, 2012.

MADALGO – Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation