The Relative Worst Order Ratio Applied to Paging

Joan Boyar

Department of Mathematics and Computer Science
University of Southern Denmark, Odense

Joint work with
Lene M. Favrholdt
Kim S. Larsen

University of Southern Denmark
Paging Problem

- **Cache**: $k$ pages
- **Slow memory**: $N > k$ pages

- **Request sequence**: sequence of page numbers
- **Fault**: page requested not in cache
- **Cost**: 1 per fault to bring page into cache
- **Goal**: minimize cost
Refinements of competitive analysis

Max/Max Ratio
[Ben-David, Borodin 94]
Compares $\mathcal{A}$ to OPT
on worst sequences of length $n$.

Random Order Ratio
[Kenyon 95]
Compares $\mathcal{A}$ to OPT
on random ordering of same sequence.
Relative Worst Order Ratio

\[ A_W(I) : A' s \text{ performance on worst permutation of } I \text{ wrt. } A \]

Intuitively: \( WR_{A,B} = \text{ worst-case } \frac{A_W(I)}{B_W(I)} \text{ on long } I \)
Relative Worst Order Ratio

[Boyar, Favrholdt 03]
Formally:

\[ \text{c}_l(A, B) = \sup \{ c \mid \exists b: \forall I: A_W(I) \geq c B_W(I) - b \} \]
\[ \text{c}_u(A, B) = \inf \{ c \mid \exists b: \forall I: A_W(I) \leq c B_W(I) + b \} . \]

If \( \text{c}_l(A, B) \geq 1 \) or \( \text{c}_u(A, B) \leq 1 \), the algorithms are comparable. Then the relative worst-order ratio \( \text{WR}_{A,B} \) is defined.
Otherwise, \( \text{WR}_{A,B} \) is undefined.
Relative Worst Order Ratio

\[ c_l(\mathbb{A}, \mathbb{B}) = \sup \{ c \mid \exists b: \forall I: \mathbb{A}_W(I) \geq c \mathbb{B}_W(I) - b \} \]
\[ c_u(\mathbb{A}, \mathbb{B}) = \inf \{ c \mid \exists b: \forall I: \mathbb{A}_W(I) \leq c \mathbb{B}_W(I) + b \} . \]

If \( c_l(\mathbb{A}, \mathbb{B}) \geq 1 \), then \( \text{WR}_{\mathbb{A}, \mathbb{B}} = c_u(\mathbb{A}, \mathbb{B}) \), and if \( c_u(\mathbb{A}, \mathbb{B}) \leq 1 \), then \( \text{WR}_{\mathbb{A}, \mathbb{B}} = c_l(\mathbb{A}, \mathbb{B}) . \)
Relative Worst Order Ratio

$c_l(A, B) \geq 1$ or $c_u(A, B) \leq 1$: One algorithm is at least as good as the other.

$WR_{A,B}$ bounds how much better.

Values of $WR_{A,B}$:

<table>
<thead>
<tr>
<th></th>
<th>minimization</th>
<th>maximization</th>
</tr>
</thead>
<tbody>
<tr>
<td>A better than B</td>
<td>$&lt; 1$</td>
<td>$&gt; 1$</td>
</tr>
<tr>
<td>B better than A</td>
<td>$&gt; 1$</td>
<td>$&lt; 1$</td>
</tr>
</tbody>
</table>
Algorithms: LRU vs. FWF

LRU – Least Recently Used
FWF – Flush When Full
Both have competitive ratio $k$.

Example sequence, $k = 5$:

$$\langle 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1, 2, 3, 4, 5, 6, 5, 4, 3, 2 \rangle$$

Total cost LRU = 8
Total cost FWF = 20
FWF vs. LRU

$I_{LRU}$ – worst ordering of $I$ for LRU

$\forall I \quad FWF_W(I) \geq FWF(I_{LRU}) \geq LRU_W(I)$

Thus, $c_1(FWF, LRU) \geq 1$ holds.
FWF vs. LRU

\[ I^n = \langle 1, 2, \ldots, k, k+1, k, \ldots, 3, 2 \rangle^n \]

\[ \text{FWF}_W(I^n) = 2kn \]

Worst ordering for LRU:

\[ \langle 2, \ldots, k, k+1, 1 \rangle^n, \langle 2, \ldots, k \rangle^n \]

\[ \text{LRU}_W(I^n) = n(k+1) + k - 1 \]

Theorem. \( WR_{\text{FWF}, \text{LRU}} \geq \frac{2k}{k+1} \)

Theorem. \( WR_{\text{FWF}, \text{LRU}} = \frac{2k}{k+1} \)
Look-Ahead

Model: $\mathbb{A}$ sees request + next $l$ requests: Look-ahead($l$)

On-line $\rightarrow$ Look-ahead($l$) $\rightarrow$ OPT

Fact 3: $k$ is still best possible competitive ratio, even with look-ahead $l$. 
Other Models of Look-Ahead

Resource-bounded look-ahead [Young 91]

Strong look-ahead [Albers 93]

Natural look-head [Breslauer 98]
Look-ahead

**LRU(ℓ):**
- Sees current page and next \( l \) pages.
- Avoids evicting pages it sees.
- Evicts l.r.u. among others in cache.

First show \( c_1(LRU, LRU(ℓ)) \geq 1 \) holds:

*Theorem.* For any sequence \( I \),
\[
LRU_W(I) \geq LRU(ℓ)_W(I).
\]
Sequence $I$. Partition into phases:

**LRU**$(\ell)$ faults $k + 1$ times per phase.

Suppose $\leq k$ distinct pages in phase $P$.

$$\langle \ldots p_1, \ldots, p, \ldots, q, \ldots, p, \ldots, p_s, p_{s+1}, \ldots \rangle$$

phase $P$; $k+1$ faults for LRU$(\ell)$

Page $p$ evicted when $q$ requested.

Least recently used not among next $\ell$. 
LRU vs. LRU(ℓ)

Case $p$ not among next $ℓ$:

$$\langle \ldots, p_1, \ldots, p, \ldots, q, \ldots, p, \ldots, p_s, p_{s+1}, \ldots \rangle$$

$P'$ has $q$ and $\geq k - 1$ distinct pages.

Phase $P$ has $\geq k + 1$ distinct pages.
LRU vs. LRU$^\ell$ 

Case $p$ not among next $\ell$:

\[
\langle \ldots p_1, \ldots, p, \ldots, q, \ldots, p, \ldots, p_s, p_{s+1}, \ldots \rangle
\]

$P'$ has $q$ and $\geq k - 1$ distinct pages.
Phase $P$ has $\geq k + 1$ distinct pages.

Case $p$ among next $\ell$:

\[
\langle \ldots p_1, \ldots, p, \ldots, q, \ldots, p, \ldots, p_s, p_{s+1}, \ldots \rangle
\]

$\geq k - 1$ distinct in $P''$; $\geq k + 1$ in $P$. 
Process \( I \) by phases.
Example sequence, \( k = 5 \) and \( \ell = 2 \):

\[
\langle 1, 2, 3, 4, 5, 6, \mid || \ 5, 7, 1, 8, 4, 2, 5, 9, 3 \rangle
\]

Reorder phase with new pages first; others in order from last phase.

\[
\langle 1, 2, 3, 4, 5, 6, \mid || \ 7, 8, 9, 1, 2, 3, 4, 5, 5 \rangle
\]

LRU faults on \( \geq \) as many as LRU(\( \ell \)).
Consider $I^n = \langle 1, 2, .., k, k + 1 \rangle^n$. $I^n$ has only $k + 1$ pages. LRU faults on every page.

Suppose $l \leq k - 1$. Whenever LRU($\ell$) faults (after first $k$ faults), it doesn’t fault on next $l$ requests.

Suppose $l \geq k$. LRU($\ell$) faults on $\leq 1$ page out of $k$.

**Theorem.** $\text{WR}_{\text{LRU,LRU}(\ell)} \geq \min\{l + 1, k\}$. 
Retrospective-LRU

Mimic the optimal algorithm, LFD.

Phases with marking:

**Basic Ideas**

- Remove marks at start of new phase.
- Mark a requested page if in LFD’s cache.
- Avoid evicting marked pages if possible.
- Within the marked/unmarked groups, evict using LRU.
- Start new phase if 2nd fault on same page.
RLRU: request $r$ to page $p$

if $p$ is not in cache then
  if there is no unmarked page then
    evict the least recently used page in cache
  else
    evict the least recently used unmarked page
if second fault on $p$ in current phase then
  unmark all pages and start a new phase with $r$
if $p$ was in LFD’s cache just before this request then
  mark $p$
else
  if $p$ is different from the previous page then
    mark $p$
RLRU – Execution

Example sequence, \( k = 5 \):

\[
\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle
\]

Total cost = 0

Cache initially empty.
Example sequence, $k = 5$:

$$\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle$$

Total cost = 1

Cache filling up.
Example sequence, $k = 5$:

$$\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle$$

Total cost = 2

Cache filling up.
Example sequence, $k = 5$:

$\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle$

Total cost = 3

Cache filling up.
Example sequence, $k = 5$:

$\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle$

Total cost = 4

Cache filling up.
Example sequence, $k = 5$:

$\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle$

Total cost = 5

Cache filling up.
Example sequence, $k = 5$:

$$\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle$$

Total cost = 6

Least recently used evicted.
Example sequence, $k = 5$:

$\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle$

Total cost = 7

Least recently used evicted. Page marked.
Example sequence, $k = 5$:

$\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle$

Total cost = 8

Least recently used unmarked evicted.
Page marked.
Example sequence, $k = 5$:

$\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle$

\[
\begin{array}{ccccc}
6 & 1 & 2 & 3 & 5 \\
\end{array}
\]

Total cost = 9

Least recently used unmarked evicted.
Page marked.
Example sequence, $k = 5$:

$\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle$

Total cost = 10

Least recently used unmarked evicted.
Page marked.
RLRU – Execution

Example sequence, $k = 5$:

$\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle$

7 1 2 3 4

Total cost = 11

Least recently used unmarked evicted.
Example sequence, $k = 5$:

$$\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle$$

Total cost = 12

Least recently used unmarked evicted.
Example sequence, \( k = 5 \):

\[
\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle
\]

Total cost = 12

No fault!
Example sequence, $k = 5$:

$$\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle$$

Total cost = 12

No fault!
Example sequence, $k = 5$:

\[\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle\]

Total cost = 12

No fault!
**RLRU – Execution**

Example sequence, $k = 5$: 

$$\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle$$

<table>
<thead>
<tr>
<th>8</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
</table>

Total cost = 12

No fault!
RLRU – Execution

Example sequence, $k = 5$:

$$\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle$$

Total cost = 13

Least recently used unmarked evicted.
Example sequence, $k = 5$:

$$\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle$$

Total cost = 14

Least recently used unmarked evicted.
Example sequence, $\ell = 5$:

$$\langle 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 7, 8, 1, 2, 3, 4, 9, 10 \rangle$$

Asymptotically, RLRU faults on 2 pages per group (regardless of ordering).

LRU faults on $\ell + 1$ pages per group.

So LRU can be a factor $\frac{\ell + 1}{2}$ worse than RLRU.
Experimental Results

Tested on a collection of traces from various applications:

- key word searches in text files
- selections and joins in Postgres
- external sorting
- various kernel operations

Trace lengths vary from 18,533 to 95,723 requests.
Cache sizes powers of two from 8 through 2048.
For higher powers, all pages can fit in cache
(for most sequences).
## Experimental Results

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\text{sort}$</th>
<th>$j_1$</th>
<th>$j_2$</th>
<th>$j_3$</th>
<th>$j_4$</th>
<th>$j_5$</th>
<th>$j_6$</th>
<th>$\text{join}$</th>
<th>$pq$</th>
<th>$xds$</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>12619</td>
<td>470</td>
<td>8177</td>
<td>4243</td>
<td>7201</td>
<td>25332</td>
<td>4596</td>
<td>7718</td>
<td>9277</td>
<td>10762</td>
</tr>
<tr>
<td></td>
<td>10736</td>
<td>468</td>
<td>8134</td>
<td>4255</td>
<td>7221</td>
<td>25326</td>
<td>4525</td>
<td>7003</td>
<td>9259</td>
<td>10709</td>
</tr>
<tr>
<td></td>
<td>14.92</td>
<td>0.43</td>
<td>0.53</td>
<td>-0.28</td>
<td>-0.28</td>
<td>0.02</td>
<td>1.54</td>
<td>9.26</td>
<td>0.19</td>
<td>0.49</td>
</tr>
<tr>
<td>64</td>
<td>10587</td>
<td>136</td>
<td>8120</td>
<td>4230</td>
<td>7135</td>
<td>25276</td>
<td>4505</td>
<td>6879</td>
<td>9185</td>
<td>10754</td>
</tr>
<tr>
<td></td>
<td>10402</td>
<td>137</td>
<td>8057</td>
<td>4239</td>
<td>7140</td>
<td>25278</td>
<td>4506</td>
<td>6838</td>
<td>9103</td>
<td>10695</td>
</tr>
<tr>
<td></td>
<td>1.75</td>
<td>-0.74</td>
<td>0.78</td>
<td>-0.21</td>
<td>-0.07</td>
<td>-0.01</td>
<td>-0.02</td>
<td>0.60</td>
<td>0.89</td>
<td>0.55</td>
</tr>
<tr>
<td>256</td>
<td>10238</td>
<td>126</td>
<td>8118</td>
<td>4213</td>
<td>7039</td>
<td>25209</td>
<td>4499</td>
<td>6793</td>
<td>8989</td>
<td>10564</td>
</tr>
<tr>
<td></td>
<td>10166</td>
<td>126</td>
<td>8057</td>
<td>4221</td>
<td>7038</td>
<td>24913</td>
<td>4492</td>
<td>6780</td>
<td>8984</td>
<td>10534</td>
</tr>
<tr>
<td></td>
<td>0.70</td>
<td>0.00</td>
<td>0.75</td>
<td>-0.19</td>
<td>0.01</td>
<td>1.17</td>
<td>0.16</td>
<td>0.19</td>
<td>0.06</td>
<td>0.28</td>
</tr>
<tr>
<td>1024</td>
<td>9618</td>
<td>126</td>
<td>5060</td>
<td>1921</td>
<td>6709</td>
<td>24024</td>
<td>4476</td>
<td>6042</td>
<td>8674</td>
<td>10190</td>
</tr>
<tr>
<td></td>
<td>9532</td>
<td>126</td>
<td>4157</td>
<td>1799</td>
<td>6674</td>
<td>23693</td>
<td>4470</td>
<td>6040</td>
<td>8607</td>
<td>10183</td>
</tr>
<tr>
<td></td>
<td>0.89</td>
<td>0.00</td>
<td>17.85</td>
<td>6.35</td>
<td>0.52</td>
<td>1.38</td>
<td>0.13</td>
<td>0.03</td>
<td>0.77</td>
<td>0.07</td>
</tr>
</tbody>
</table>
Experimental Results
Other Results with Relative Worst Order Ratio

1. Bin Packing: Worst-Fit better than Next-Fit.
2. Dual Bin Packing: First-Fit better than Worst-Fit.
3. Scheduling – minimizing makespan: Post-Greedy better than putting all jobs on fast machine, for two related machines.
4. Bin Coloring: Greedy better than keeping only one open bin.
5. Proportional Price Seat Reservation: First-Fit better than Worst-Fit.