# Fast Sequence Segmentation using Log-Linear Models

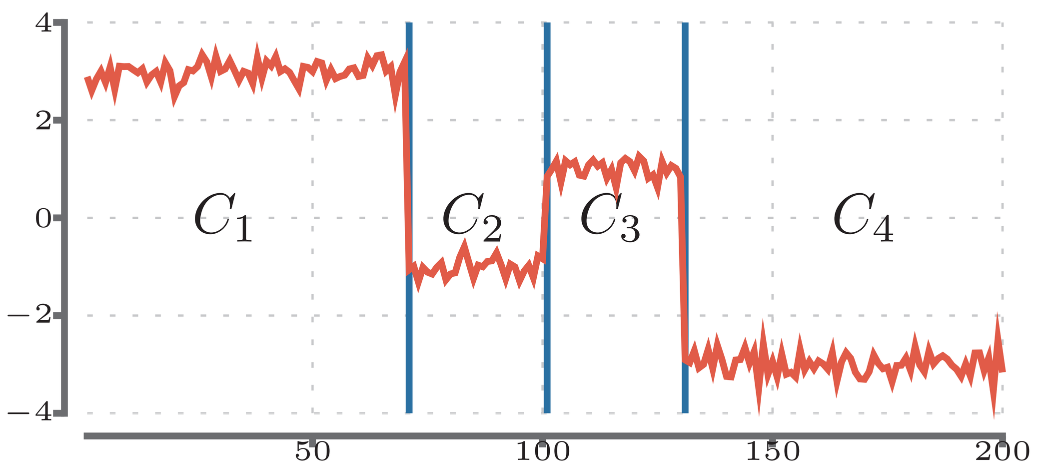
NIKOLAJ.TATTI@AALTO.FI
Aalto University, Helsinki Institute of Information Technology

Aalto University

HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

## Segmentation

Given sequence $s$, and a number $K$ divide $s$ into $K$ cohesive segments




## Dynamic Program

If score is additive, optimal solution can be found by dynamic program

**foreach** $k = 2, \ldots, K$ **do**
  **foreach** $i = 1, \ldots, N$ **do**
    **foreach** $j = 1, \ldots, i$ **do**
      $C \leftarrow opt(k-1, j-1)$ and $(i, j)$;
      **if** $sc(C) < sc(O)$ **then** $O \leftarrow C$;
      ;
    $opt(k, i) \leftarrow O$;

$O(KN)$ space and $O(KN^2)$ time

## Speed-up

Do not visit every $j$,
instead keep list of candidates $P$

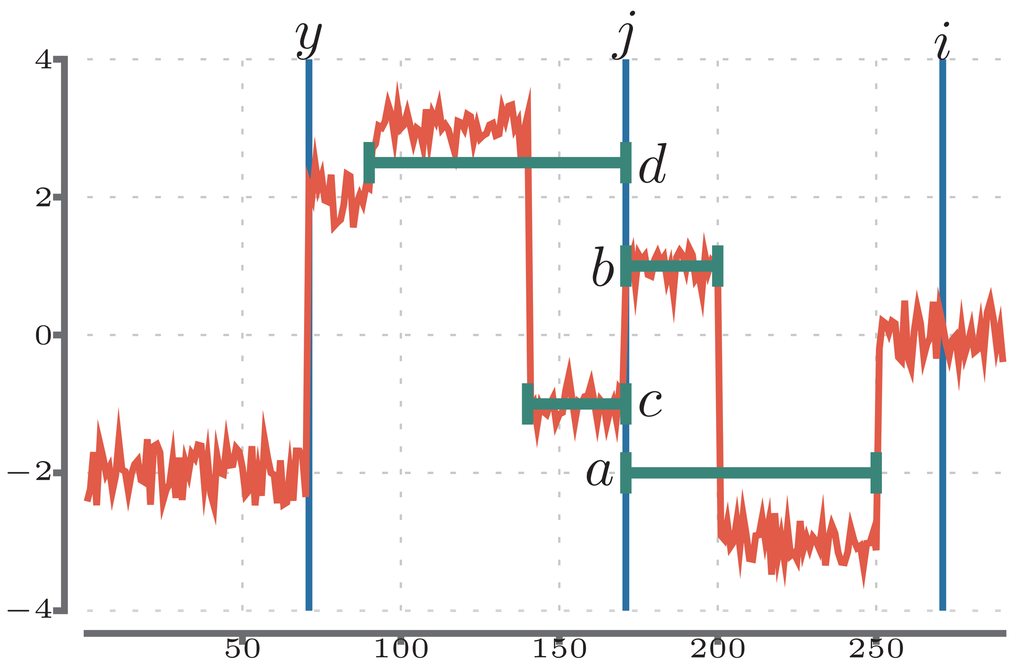
Whenever possible, trim $P$

**foreach** $k = 2, \ldots, K$ **do**
  $P \leftarrow \emptyset$;
  **foreach** $i = 1, \ldots, N$ **do**
    add $i$ into $P$;
    **foreach** $j \in P$ **do**
      compute segmentation;
      **if** *$j$ is guaranteed to be suboptimal* **then** remove $j$;
      ;
    $opt(k, i) \leftarrow C$;

## Sufficient Condition

Segment $s[1, i]$ with $K = 3$ segments

- $[j, i]$ is a candidate for the last segment
- $[1, y-1], [y, j-1]$ is the optimal 2-segmentation for $s[1, j-1]$




Define:
$left(j, i) = [a, b]$, where

$$a = \min_{j \le x \le i} (\text{average of } s[j, x])$$

$$b = \max_{j \le x \le i} (\text{average of } s[j, x])$$

and $right(k, j-1) = [c, d]$, where

$$c = \min_{y \le x < j} (\text{average of } s[x, j-1])$$

$$d = \max_{y \le x < j} (\text{average of } s[x, j-1])$$

$y$ is the starting index of the last segment in $k$-segmentation for $s[1, j-1]$

THEOREM: If $left(j, i)$ and $right(k-1, j-1)$ overlap, then $j$ cannot be the starting index for the last segment of optimal $k$-segmentation for $s[1, i]$

If the theorem holds for $i$, it also holds for $i' > i$.

- keep intervals for every $j$ in $P$.
- delete $j$ from $P$ as soon overlap occurs.

## Computing Intervals

Code for updating intervals:

**foreach** $k = 2, \ldots, K$ **do**
  $P \leftarrow \emptyset$;
  **foreach** $i = 1, \ldots, N$ **do**
    add $i$ into $P$;
    **foreach** $j \in P$ **do**
      compute segmentation;
      compute $left(j, i)$;
      **if** *$left(j, i)$ and $right(k-1, j-1)$ overlap* **then** remove $j$ from $P$;
      ;
    $opt(k, i) \leftarrow C$;
    compute $right(k, i)$;

Left interval is easy: let $\mu$ = average of $s[j, i]$.

$$left(j, i) = [\min(a, \mu), \max(b, \mu)]$$

Right interval is harder:

- $i$ goes into different direction
- optimal segmentation is needed

## PAV algorithm

Compute right interval with PAV algorithm

- online algorithm, input a stream of numbers, $x_1, \ldots$
- at $i$th point returns the largest average
- amortized constant time, linear space

At $i$th point maintain a $x_1, \ldots, x_i$ arranged into blocks, each block has higher average than previous. The last block has the highest average.

Update step:

add new point as a single block;
**while** *violating monotonicity* **do**
  merge last two blocks;

## Right Intervals

Keep blocks $pav(j, i)$ for $s[j, i]$, for $j \in P$.

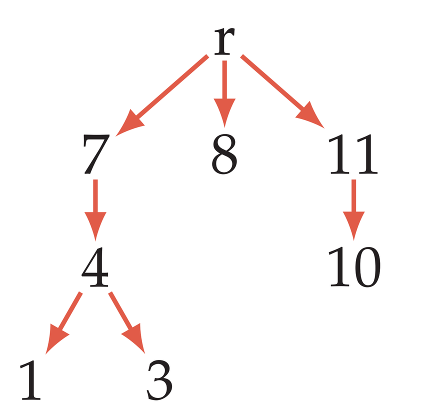
When optimal last segment is known, say $j^*$, compute the right interval from $pav(j^*, i)$.

Lists require quadratic space.
Can be rearrange into a tree:

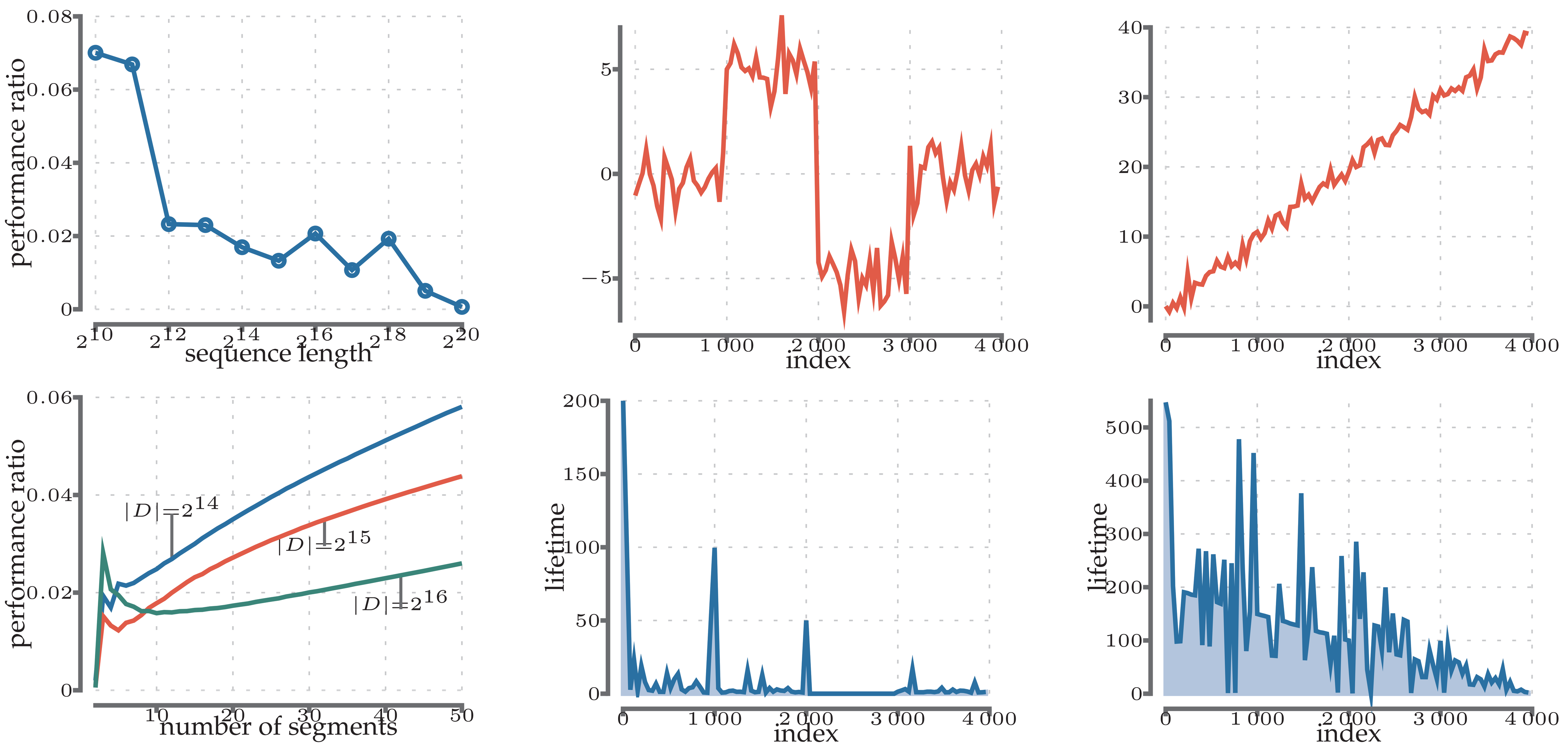
Sequence $s = (2, 0, 1, 2, 1, 1, 9, 2, 5, 0, 1)$
Potential candidates $P = (1, 3, 8, 10, 11)$

Blocks (start indices):
$pav(1, 11) = (1, 4, 7)$
$pav(3, 11) = (3, 4, 7)$
$pav(8, 11) = (8)$
$pav(10, 11) = (10, 11)$
$pav(11, 11) = (11)$

Tree:




## Experiments




performance ratio = total number of score comparisons, normalized between 0 and 1
lifetime = how many iterations index is in $P$