Backward Weighted Coding

Aharon Fruchtman | Yoav Gross | Shmuel T. Klein | Dana Shapira

Static & dynamic techniques

- Static coding
  - Prelude: Probability distribution.
- Dynamic backward **looking coding** (b-adp)
  - Prelude: Negligible, no description of the model is needed.
- Dynamic forward **looking coding** (f-adp)
  - Prelude: Exact frequencies of the elements.

Weighted coding

Given a file $T = T[1, n]$ of n characters over an alphabet Σ .

Define a general weight $W(g, \sigma, \ell, u)$:

- $g: [1, n] \longrightarrow \mathbb{R}^+$
- $\sigma \in \Sigma$
- $1 \leq \ell \leq u \leq n$ boundaries of an interval.

$$W(g, \sigma, \ell, u) = \sum_{\substack{\ell \leq j \leq u \\ T[j] = \sigma}} g(j) \, .$$

Weighted coding – example

$$W(g, \mathrm{b}, 2,8) = \sum_{\substack{2 \leq j \leq 8 \\ T[j]=\mathrm{b}}} g(j) =$$

i	1	2	3	4	5	6	7	8	9	10
T	d	a	b	c	d	c	a	b	b	a
$g(i)$	1	4	2	8	3	5	6	3	1	7

Weighted coding – example

$$W(g, \mathrm{b}, 2{,}8) = \sum_{\substack{2 \le j \le 8 \\ T[j]=\mathrm{b}}} g(j) = 5$$

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
T	d	a	**b**	c	d	c	a	**b**	b	a
$g(i)$	1	4	**2**	8	3	5	6	**3**	1	7

Generalization

The constant function: $\mathbb{1} \equiv g(i) = 1$ for all i.

- Static coding:

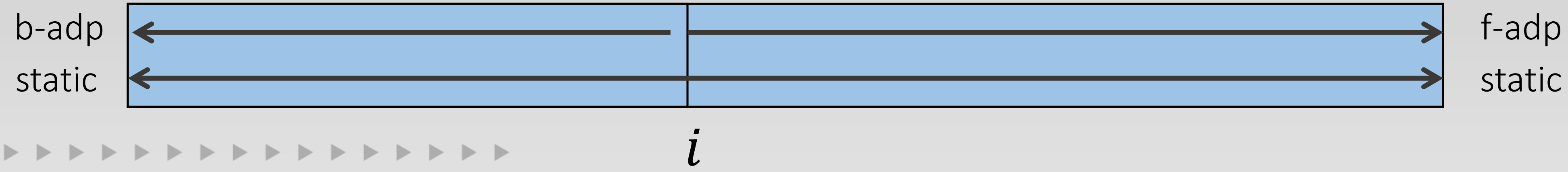
$$W(\mathbb{1}, \sigma, 1, n)$$

- Backward adaptive coding:

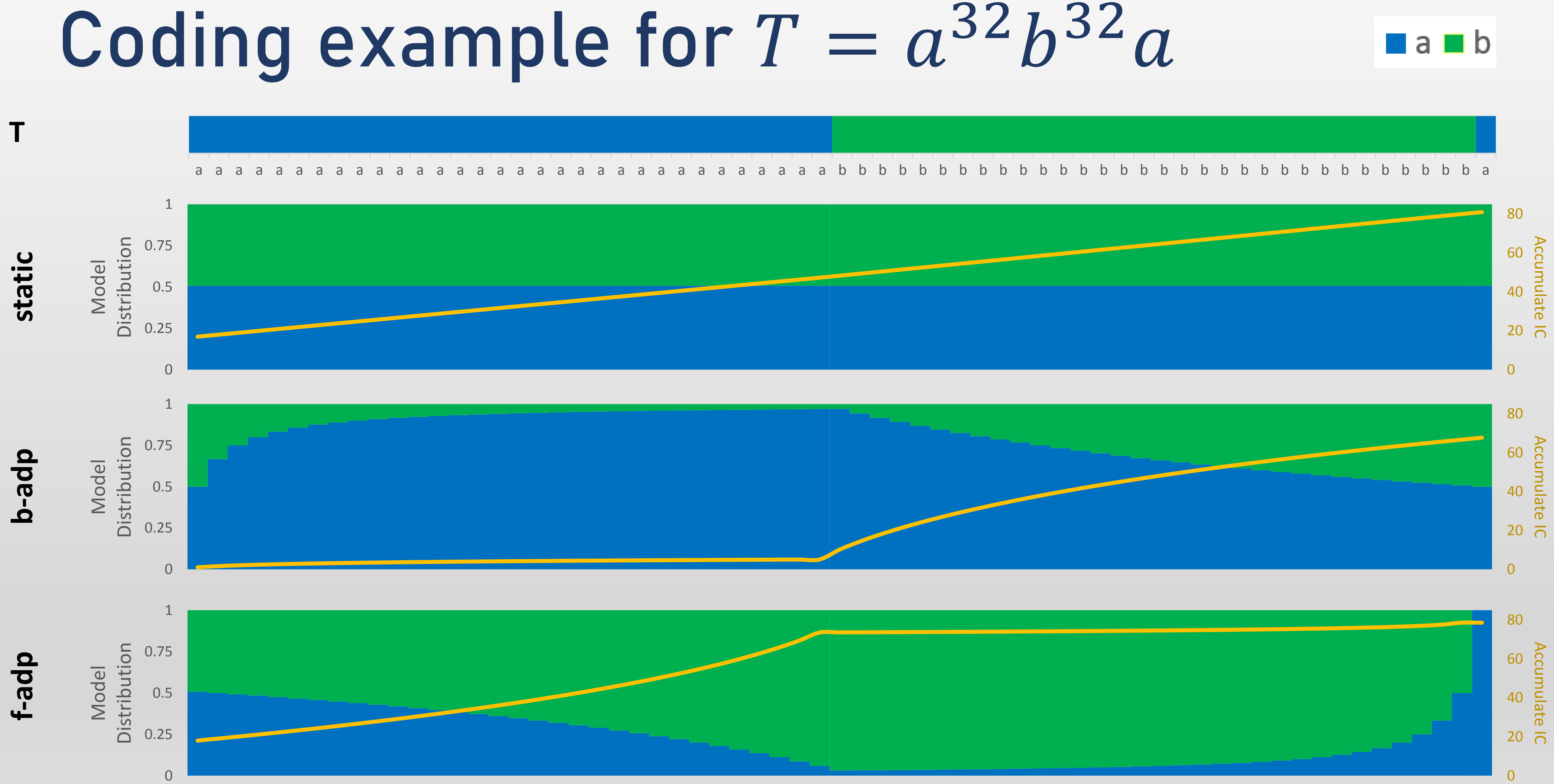
$$W(\mathbb{1}, \sigma, 1, i-1)$$

- Forward adaptive coding (f-adp):

$$W(\mathbb{1}, \sigma, i, n)$$



Coding example for $T = a^{32}b^{32}a$

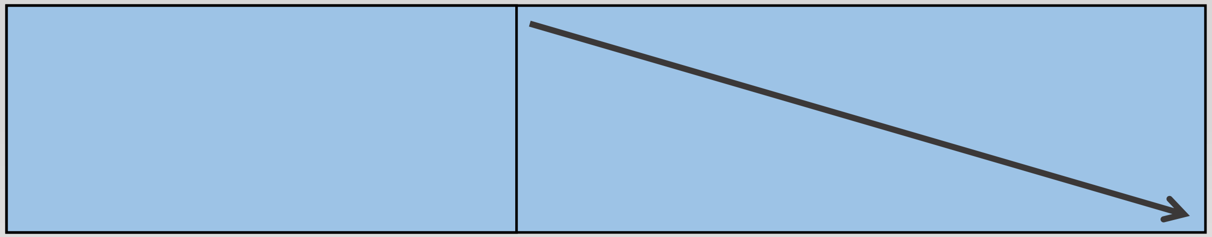


Forward weighted coding

Relative to position i, using a decreasing function g:

$$W(g, \sigma, i, n) = \sum_{\substack{i \le j \le n \\ T[j]=\sigma}} g(j)$$

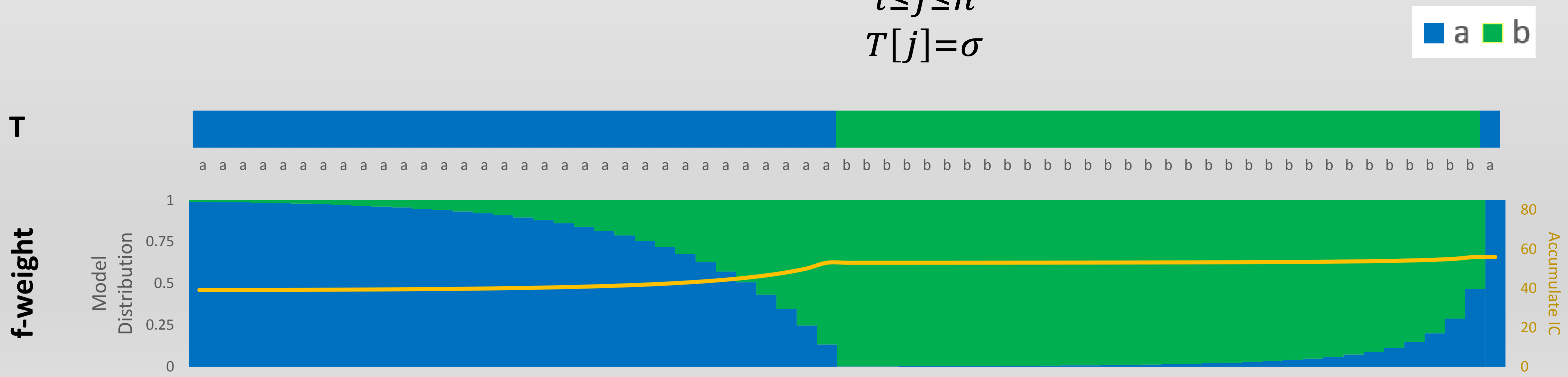
- Increased consideration to closer locations in front.
- Heavy prelude
  - Exact weights of the elements.



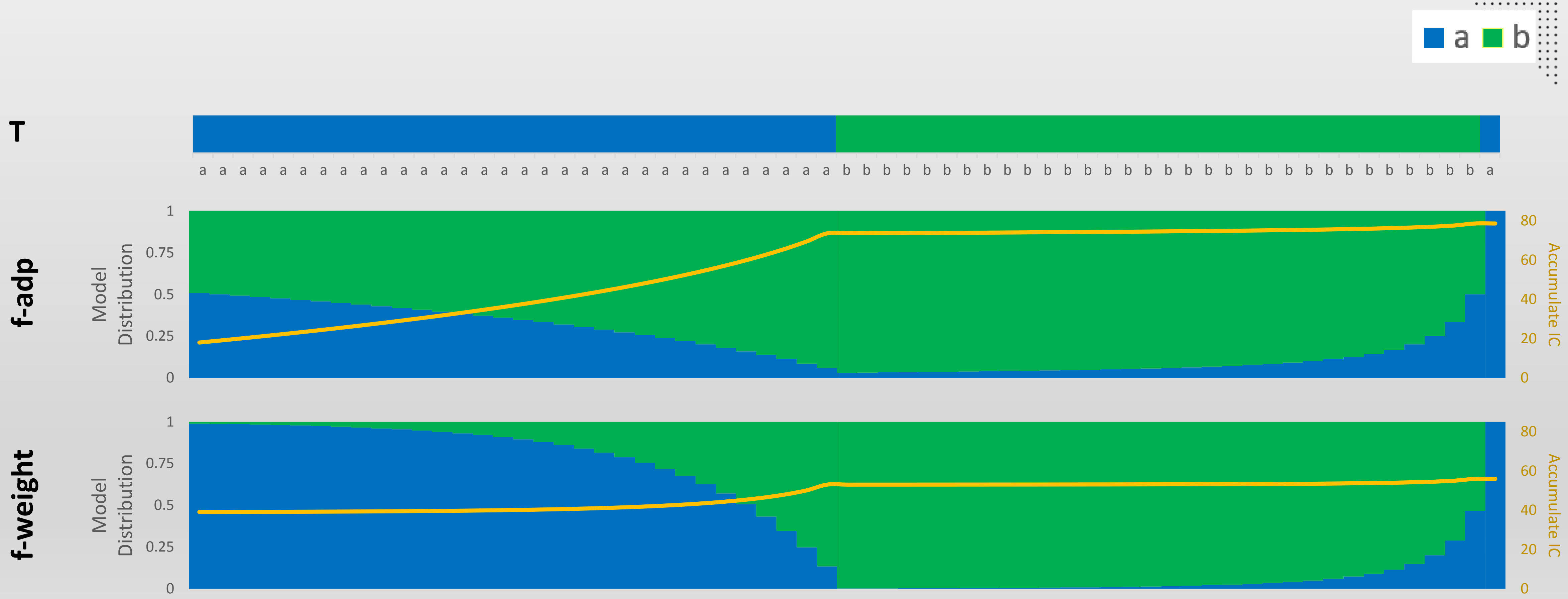
f-weight (example)

On our example we applied: $g(i) = 1.15^{n-i}$

$$W\left(1.15^{n-i}, \sigma, i, n\right) = \sum_{\substack{i \le j \le n \\ T[j]=\sigma}} 1.15^{n-i}$$



F-weight (example)

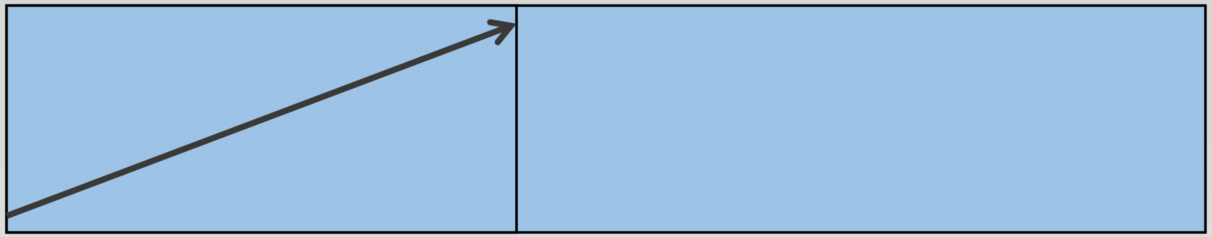


Backward weighted coding

Relative to position i, using an **increasing** function g:

$$W(g, \sigma, 1, i-1) = \sum_{\substack{1 \le j \le i-1 \\ T[j]=\sigma}} g(j)$$

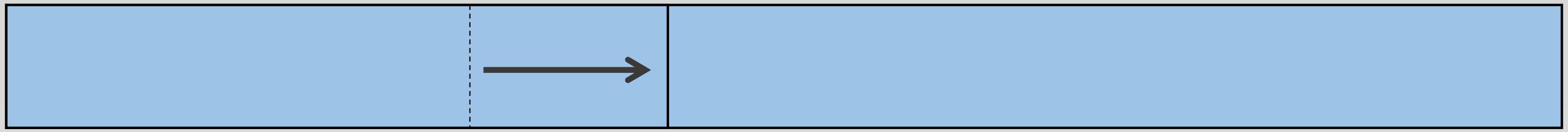
- Increased consideration to closer locations from behind.
- Negligible header



Sliding window

- An active window of size k, determined by the interval $[i-k, i-1]$ for position i.
- Ignores the beginning of the input file.

$$g(j) = \begin{cases} 1 & i-k \leq j < i \\ 0 & \text{otherwise} \end{cases}$$



Division by 2

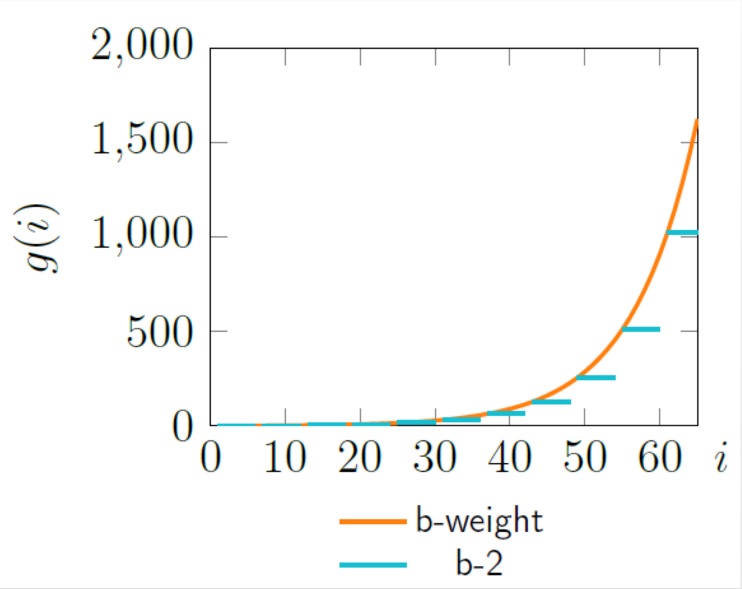
- Nelson (1996) rescaled the weights from time to time, to make sure that each character frequency may be represented by 16 bits.
- He noted that **division by 2** also improves the quality of the compression.
- Not completely ignore the beginning, but rather gives them less importance than closer ones.

b-2

- b-2 is a different backward method based on the division by 2.
- Uses some fixed number k of characters between the division points, rather than letting this number be controlled by technical issues.

$$g_{\mathrm{b-2}}(i) = 2^{\left\lfloor \frac{i-1}{k} \right\rfloor} = \begin{cases} 1 & 1 \le i \le k \\ 2g_{\mathrm{b-2}}(i-k) & \text{otherwise} \end{cases}$$

b-weight



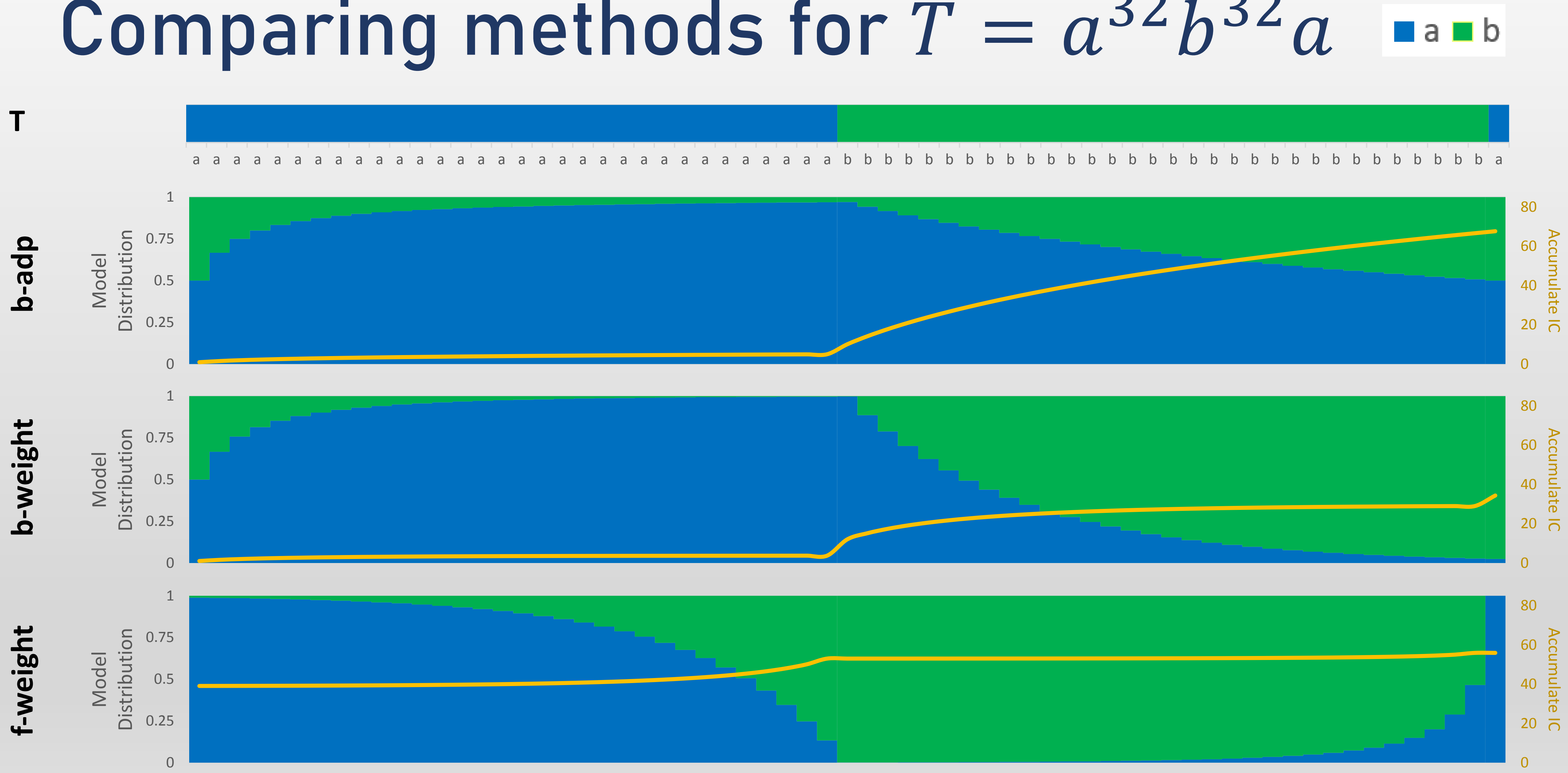
- Refined version of $g_{\mathrm{b-2}}$, rather than using the same values within a block.
- A fixed ratio between adjacent indices. For $i \geq 1$

$$g_{\mathrm{b-weight}}(i) = \left(\sqrt[k]{2}\right)^{i-1}$$

- The fixed ratio of 2 between blocks is also maintained:

$$g_{\mathrm{b-weight}}(i+k) = 2 \cdot \left(\sqrt[k]{2}\right)^{i-1} = 2 g_{\mathrm{b-weight}}(i)$$

Comparing methods for $T = a^{32}b^{32}a$



Running example – summary

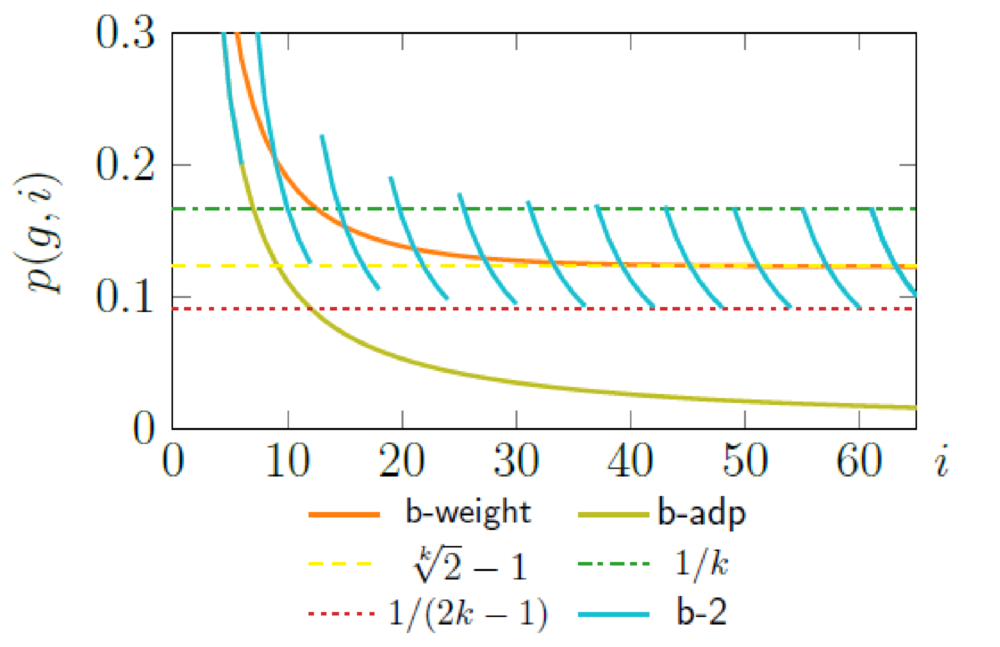
- Storage requirements of the encoding methods on $T = a^{32}b^{32}a$:

	Header			*H*	*Total*
	a	b	bps		
static	33	32	0.246	1.000	1.246
b-adp	–	–	–	1.041	1.041
f-adp	33	32	0.246	0.948	1.194
f-weight	58115	664	0.600	**0.260**	0.860
b-weight	–	–	–	0.530	**0.530**
b-2	–	–	–	0.536	0.536

Choosing the constant k

- Choosing the constant k for **b-2** and **b-weight** is a trial-and-error process.
- A trade-off between processing time and compression performance.
- Our experiments indicate that preprocessing even a small prefix of the file suffices to find satisfying values of k.

Analysis



- Evaluate the ratio, for $2 \leq i \leq n$:

$$p(g, i) = \frac{g(i)}{\sum_{j=1}^{i-1} g(j)}$$

- For **b-adp**, $p(\mathbb{1}, i) = \frac{1}{i-1} \longrightarrow 0$
- For **b-2** and given constant k, for i large enough, $p(g_{\mathrm{b-2}}, i) \in \left[\frac{1}{2k-1}, \frac{1}{k}\right]$
- For **b-weight** and given constant k, $p\left(g_{\mathrm{b-weight}}, i\right) \longrightarrow \sqrt[k]{2} - 1$

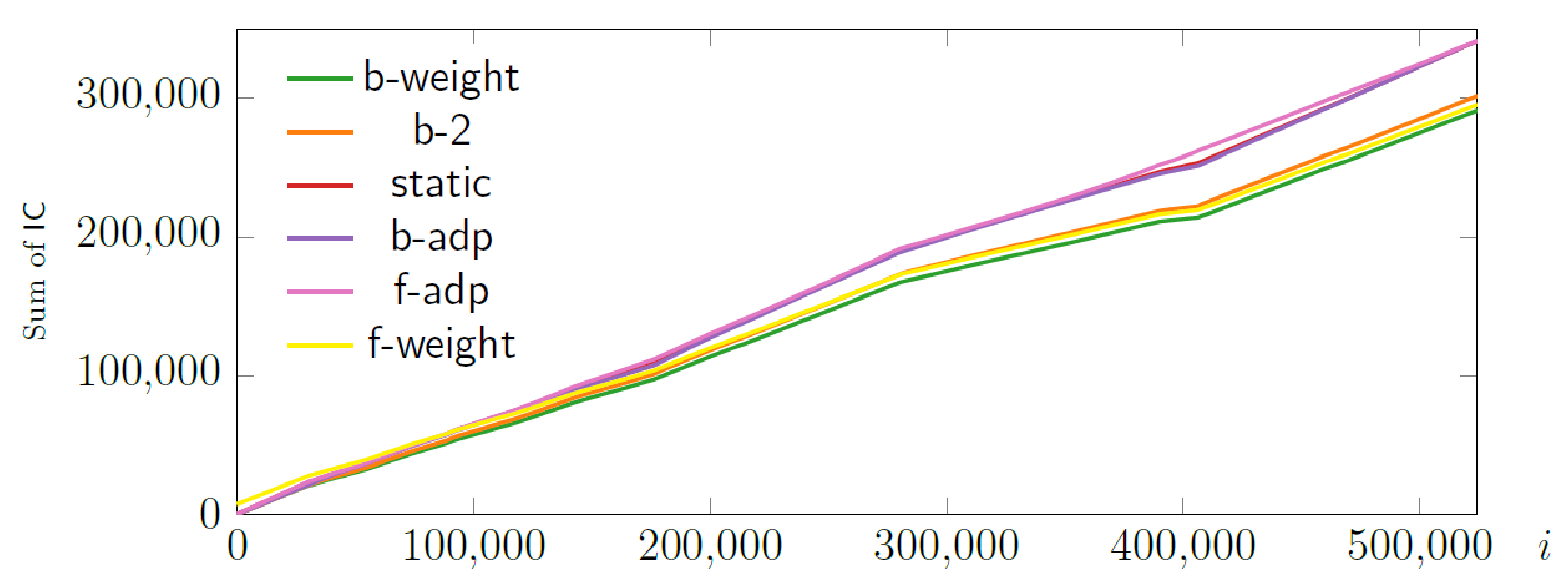
Experimental Results

- Compression performance (%) of different methods using arithmetic coding:

	H_0	static	b-adp	f-adp	b-2 (k)		b-weight (k)	
SOURCES	69.21	69.21	69.21	69.21	64.38	(3,104)	**62.58**	(318)
XML	65.37	65.38	65.38	65.38	65.05	(35,840)	**64.93**	(6,314)
DNA	24.78	24.77	24.78	24.77	24.74	(141,312)	**24.44**	(85)
ENGLISH	56.61	56.61	56.61	56.61	56.24	(26,112)	**56.06**	(3,775)
PITCHES	70.41	70.42	70.42	70.42	57.55	(385)	**46.05**	(32)
PROTEINS	52.44	52.44	52.44	52.44	52.08	(34,304)	**51.59**	(407)

Experimental Results

- Compression efficiency as a function of progress on a prefix of sources of size 512KB:



Experimental Results – PPM

- PPM – Prediction by Partial Matching.
- Compression performance (%) of different methods using PPM:

	r	H_r	PPM	b-2 (k)		b-weight (k)	
SOURCES	2	37.65	37.93	33.87	(422)	**29.66**	(13)
	3	27.68	28.97	26.81	(296)	**25.36**	(8)
XML	2	25.09	25.24	22.76	(832)	**21.88**	(81)
	3	16.53	17.21	**15.80**	(472)	15.86	(42)
DNA	2	24.06	24.06	23.99	(4,656)	**23.90**	(129)
	3	24.00	24.00	23.95	(3,406)	**23.90**	(216)
ENGLISH	2	36.53	36.62	35.90	(1,505)	**35.55**	(188)
	3	29.83	30.27	**29.73**	(800)	29.89	(110)
PITCHES	2	51.74	52.23	48.21	(306)	**44.73**	(14)
	3	**43.21**	47.81	46.13	(252)	47.81	–
PROTEINS	2	51.82	51.84	49.93	(160)	**49.82**	(76)
	3	50.43	50.70	**48.72**	(126)	49.14	(72)

Thank you!