List Update for Data Compression

Alex López-Ortiz, University of Waterloo

Reza Dorrigiv, Samsung Research Labs; Shahin Kamali, U. Waterloo; Susana Ladra, U. A Coruña; Diego Seco, U. Concepción;

List update

List update

Problem definition:

- Set of items stored in a sequential linked list.
- Items are requested in an on-line fashion, i.e. wd do not know the sequence of requests in advance
- After each access the list can be rearranged

List update

Problem definition (cont.):

- Cost of retrieving item is position in the list
- It can be moved forward for free
- Any other element can be swapped with an immediate neighbor as many times as desired at a cost of 1
- GOAL: Devise the best ordering moves for the list to minimize access_cost + reordering costs

List update

Basic rearranging schemes:

- MTF: move last accessed item to the front of the list
- Transpose: swap last accessed item with one preceding it
- FC arrange by frequency count (so far)
- Timestamp: move item past "stale" ones
- MF_k: move forward to 1/*k*-th from the front

List update

Traditionally been studied under the competitive ratio [Sleator & Tarjan 1985]:

Compare cost against offline optimum that knows the sequence of requests in advance

$$\sup_{\forall I} \frac{\#\text{ cost of online algorithm on input } I}{\#\text{ cost of offline optimum on input } I}$$

List update

Known results under ST model :

- MTF is best
- Randomized BIT is “better” (not really)
- Transpose is bad (not really)
- Lookahead has no effect on worst case behaviour (not really)

Locality of reference

- LU works best when there is locality of reference [Dorrigiv&L-O,Albers&Lauer] DLAL
- measures performance using a refined "competitive ratio" measure based on non-locality of reference
- Cost proportional to amount of non-locality

List Update for Data Compression

Bentley et al. (1986) observed that List Update can be used for data compression:

- Initialize list to alphabet in some predefined order (e.g. alphabetical)
- For each letter in the text:
  access letter in linked list
  output index of position in list (self-encoded)
  move letter to front (MTF)

List update

Bentley et al.:

- LU/MTF for characters often superior to Huffman

- LU/MTF $\leq$ 2 static Huffman

(using Elias-A self-encoding) in the worst case

- Can also be used in words instead of chars. Far superior to Huffman

List update

Albers & Mitzenmacher:

$$LU/MTF \leq 1+H(S)+2 \log(1+H(s))$$

in the worst case, where H(s)=size of Huffman encoding (using Elias-B self-encoded numbers)

Ditto for LU/TS.

Experiments show TS is better than MTF

Burrows-Wheeler Transform (BWT)

- Transforms text to a more regular form:

```
…yyyyyeteelsfydndeabwb,tflttttereenstdcmnred,trgtsol
dsbtdaalhhhhhhhhhhhhhhhhhohhhhhhhhhhhhhhhhhhhhhhhhhh
hhaesnesl]s,tyenrsltssdwderrrrrnneyyffnndee,gneek…
```

(sample from the BWT of an actual text)

- Ideally suited for LU compression

BWT

- One choice fits all: MTF is best
- Possible to do better via selection of parameter, e.g. use Timestamp(α) for 0≤ α ≤ 1
- Optimize for α by trying several options, select best [Dorrigiv&L-O'08]

LU+BWT

- MTF developed for costs of ST model: cost of item with position i in the list is i

- The "cost" of an LU algorithm with BWT is #bits to write position i in the list, i.e. $\log(i)$

LU+BWT

Possibly an important distinction:

- MF_k is 4 competitive in ST model (k=2)
- MF_k is $\log(m)$ competitive in the compression model, where m is the size of the list

LU for compression

Theorem [Kamali, L-O] MTF is 2 competitive in the restricted compression model

	book1						
	Free	Paid	AC	STD	MRM	CC_linear	CC_log
MTF	9,002,657	0	9,771,428	9,771,428	9,771,428	9,771,428	5,155,343
TS	3,215,445	[illegible]	[illegible]	[illegible]	8,326,983	8,326,983	4,653,983
CB	907,829	[illegible]	[illegible]	[illegible]	52,941,414	**3,216,242**	**2,542,883**
RCB	2,415,021	[illegible]	[illegible]	[illegible]	**7,373,290**	7,373,290	4,008,491

	[illegible]						
	Free	[illegible]	[illegible]	[illegible]	MRM	CC_linear	CC_log
MTF	6,033,483	[illegible]	[illegible]	[illegible]	6,410,592	6,410,592	2,669,833
TS	2,324,979	0	5,788,482	**5,788,482**	5,788,482	5,788,482	2,479,925
CB	967,786	[illegible]	2,271,967	359,792,280	32,919,516	**2,271,967**	**1,384,551**
RCB	2,363,356	24,002,408	5,141,341	29,143,749	**5,141,341**	5,141,341	2,128,293

	progc						
	Free	Paid	AC	STD	MRM	CC_linear	CC_log
MTF	645,069	0	684,680	684,680	684,680	684,680	281,197
TS	262,363	0	632,842	**632,842**	632,842	632,842	264,907
CB	78,196	26,575,729	223,937	26,799,666	3,010,434	**223,937**	**129,079**
RCB	159,749	1,580,853	529,991	2,110,844	**529,991**	529,991	213,139

	trans						
	Free	Paid	AC	STD	MRM	CC_linear	CC_log
MTF	1,531,585	0	1,625,280	1,625,280	1,625,280	1,625,280	656,437
TS	651,001	0	1,548,988	**1,548,988**	1,548,988	1,548,988	634,037
CB	129,422	69,476,148	380,704	69,856,852	7,339,262	**380,704**	**264,085**
RCB	485,861	4,937,879	1,344,826	6,282,705	**1,344,826**	1,344,826	522,627

From Kamali, Ladra, Lopez-Ortiz, Seco [DCC 2013]

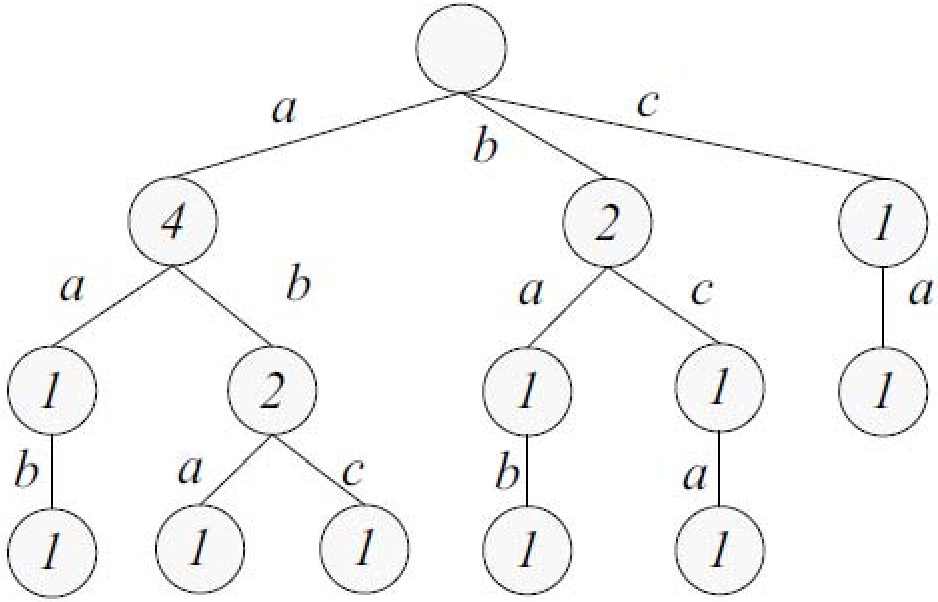
Context-Based List Update

- Idea: keep track of frequency of all contexts of length at most c seen so far
- Find largest matching context
- Encode the option according to frequencies observed

Context-Based List Update

Example string: buxu*cdux*

1. After outputting *c*, to encode *d*
2. Say we have seen contexts c, uc, and xuc, but not uxuc.
3. Then rearrange the list according to frequencies of all possible continuations of xuc (largest seen context)
4. Output index of *d* in rearranged list

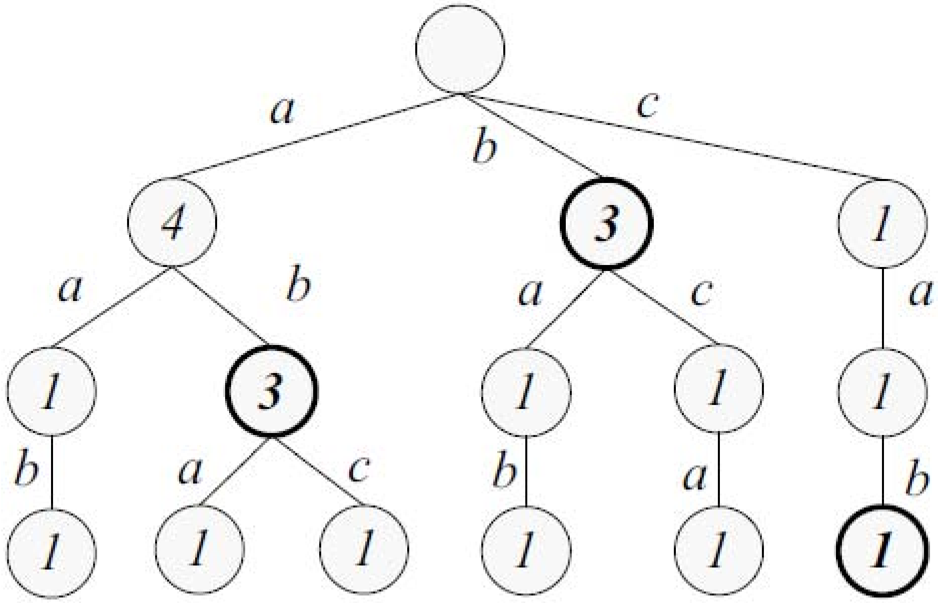


	a	b	c
ca	0	0	0
a	1	2	0
ε	4	2	1

$L =$ | b | a | c |

$\sigma = a\ a\ b\ a\ b\ c\ a\ \ldots\ b\ a$

$C = 3$ for a sequence $\sigma = aababcaba$



	a	b	c
ab	1	0	1
b	1	0	1
ε	4	3	1

a	c	b

$L = [\,a \;\; c \;\; b\,]$

$$\sigma = a\ a\ b\ a\ b\ c\ a\ b\ \ldots\ a$$

File	Size (bytes)	Before BWT					After BWT				
		MTF	CB	CB′	RCB	RCB′	MTF	CB	CB′	RCB	RCB′
bib	111261	95.69	**29.78**	30.47	70.44	72.16	30.49	34.04	36.03	32.90	32.24
book1	768771	83.82	**34.15**	35.75	63.99	65.97	35.74	38.66	40.22	36.37	36.21
book2	610856	84.35	**29.97**	30.54	65.00	65.39	31.14	34.08	35.87	32.32	32.20
geo	102400	104.91	76.69	80.46	99.43	104.37	50.78	47.87	51.79	**47.13**	48.53
news	377109	88.50	**35.05**	35.72	68.31	69.01	36.21	39.85	43.16	38.25	38.47
obj1	21504	89.99	59.38	57.39	80.40	76.11	**43.75**	46.02	49.04	45.38	44.66
obj2	246814	101.68	36.72	34.81	88.20	79.39	**28.06**	30.29	32.49	29.34	29.25
paper1	53161	86.79	**33.64**	34.21	65.11	66.82	34.70	39.44	41.93	37.68	37.08
paper2	82199	84.47	**33.50**	34.62	62.83	65.35	34.86	38.43	41.06	36.52	36.35
pic	513216	23.21	**19.54**	20.14	21.55	21.78	20.08	19.77	21.07	19.60	19.84
progc	39611	88.74	34.46	**34.34**	66.28	66.28	35.04	40.01	42.20	38.48	37.23
progl	71646	77.01	26.08	**25.71**	58.15	57.58	26.31	29.29	31.36	28.02	27.80
progp	49379	81.09	26.32	**25.90**	61.23	59.90	26.00	29.20	30.91	28.05	27.70
trans	93695	87.58	24.35	24.31	65.63	65.25	**24.12**	26.92	28.76	26.02	25.78

Table 2: Compression percentage of text files of the Calgary Corpus using different list-update algorithms. We use bold type to highlight the best values for each file.

Improvements on BWT

- Can we improve the BWT as well?
- Online algorithms with advice: theoretical model with access to an Oracle
- Goal: minimize the number of bits of advice from the oracle while obtaining (near-)optimal online performance
- Introduced by [Královič et al. in 2009] as a purely theoretical study of online algorithms

From highly theoretical to practical

- IDEA: Compression is semi-offline
- At compression time we can do several passes (e.g. Huffman requires two passes over the input)
- At decompression time however *we must* operate on line since we do not have the entire uncompressed input

Advice to the decompressor

- Encode the answers “from the future” to the decompressor in the preamble of the compressed file
- Whenever decompressor asks a question to the Oracle, we read the answer from the preamble

Better LU for compression

- Introduced BIB which
  - Divides input into blocks
  - Behaves as one of TimeStamp or MTF within each block
  - This is controlled by a single bit of advice per block

Start file name file name	original file size (bytes)	MTF	TS	BIB	block Size	compressed file size (bytes)	advice cost (bits)
Calgary Corpus							
bib	111261	30.5013	32.3195	**30.1948**	117	33595	964
book1	768771	35.7117	34.6887	**34.1462**	39	262506	19724
book2	610856	31.1388	31.4832	**30.5859**	507	186836	1222
geo	102400	79.251	78.4229	**77.8457**	211	79714	501
news	377109	36.2137	38.6721	**35.6995**	38	134626	9935
obj1	21504	57.2359	59.8726	**56.5895**	46	12169	479
obj2	246814	37.9043	41.9093	**37.8098**	121	93320	2053
paper1	53161	34.7191	37.6855	**34.388**	59	18281	913
paper2	82199	34.869	36.0369	**34.2303**	88	28137	948
paper3	46526	37.7724	39.7176	**37.076**	52	17250	906
paper4	13286	41.3367	44.6937	**40.9303**	34	5438	402
paper5	11954	42.3624	46.863	**42.2118**	17	5046	713
paper6	38105	35.2552	38.8558	**35.1371**	84	13389	467
pic	513216	20.156	19.5808	**19.5797**	519	100486	1008
progc	39611	35.0711	38.5247	**34.9221**	85	13833	480
progl	71646	26.3295	29.4308	**26.2974**	221	18841	340
progp	49379	**26.0313**	30.2193	26.0394	6030	12858	34
trans	93695	24.1176	28.6867	**24.0984**	475	22579	215
Canterbury Corpus							

Conclusions

- Theoretical study of list update for compression just beginning
- Proof of 2-competitiveness for MTF in compression model
- New best text compressor in general
- New best BWT-based text compressor