Compression de données par énumeration de sous-chaînes

Danny Dubé

Université Laval
Québec, Qc, Canada

Séminaire départemental — 1er mars, 2011

Lossless Data Compression via Substring Enumeration

Danny Dubé **Vincent Beaudoin**

Université Laval
Quebec City, Quebec, Canada

DCC'10 — Snowbird, Utah, USA — March 25, 2010

Basic idea

Compression of $\mathbf{D} \in \{0,1\}^+$ using substring enumeration, where $N = |\mathbf{D}|$:

For l **:=** 1 **to** N **do**
  For every distinct l-bit substring w of $\mathbf{D}$ **do**
    Send number of occurrences of w in $\mathbf{D}$

$O(N^2)$ numbers to send!

Definition of substring

Substring w *occurs* at *position* p in $\mathbf{D}$ if:

$$\exists\, u \in \{0,1\}^*,\ v \in \{0,1\}^\infty.\ |u| = p < N \quad \text{and} \quad u\,w\,v = \mathbf{D}^\infty.$$

Notation: C_w is the *number* of occurrences of w in $\mathbf{D}$.

$C_w =$ number of positions where w occurs.

Enumeration example (1)

From the compressor's point of view:

$\mathbf{D} = 01000001$

Length	Substrings							
1	6×0						2×1	
2	4×00				2×01		2×10	
3	3×000			1×001	2×010		1×100	1×101
4	2×0000		1×0001	1×0010	1×0100	1×0101	1×1000	1×1010
5	1×00000	1×00001	1×00010	1×00101	1×01000	1×01010	1×10000	1×10100
6	1×000001	1×000010	1×000101	1×001010	1×010000	1×010100	1×100000	1×101000
7	1×0000010	1×0000101	1×0001010	1×0010100	1×0100000	1×0101000	1×1000001	1×1010000
8	1×00000101	1×00001010	1×00010100	1×00101000	1×01000001	1×01010000	1×10000010	1×10100000

Enumeration example (2)

From the decompressor's point of view:

$N = ?$

Knowing: $N \in Naturals$

Receive $N = 8$.

Enumeration example (3)

From the decompressor's point of view:

$N = 8$

Length	Substrings
1	?×0 ?×1
...	

Knowing: $0 \leq C_0 \leq 8$

Receive $C_0 = 6$.

Enumeration example (4)

From the decompressor's point of view:

$N = 8$

Length	Substrings
1	6×0 $?\times 1$
...	

Knowing: $C_0 + C_1 = 8$

Receive nothing; just conclude $C_1 = 2$.

Enumeration example (5)

From the decompressor's point of view:

$N = 8$

Length	Substrings			
1	6×0		2×1	
2	$?\times 00$	$?\times 01$	$?\times 10$	$?\times 11$
...				

Naïvely knowing: $0 \leq C_{00} \leq 6$

But $C_{00} = 0$ implies $C_{01} = 6$.

...

But $C_{00} = 3$ implies $C_{01} = 3$.

So, knowing: $4 \leq C_{00} \leq 6$

Receive $C_{00} = 4$.

Enumeration example (6)

From the decompressor's point of view:

$N = 8$

Length	Substrings			
1	6×0		2×1	
2	4×00	$?\times 01$	$?\times 10$	$?\times 11$
...				

Knowing: $C_{00} + C_{01} = C_0$

Receive nothing; just conclude $C_{01} = 2$.

Enumeration example (7)

From the decompressor's point of view:

$N = 8$

Length	Substrings			
1	6×0		2×1	
2	4×00	2×01	?$\times 10$	?$\times 11$
...				

Knowing: $0 \leq C_{10} \leq C_1$

Really?

Knowing also: $C_{00} + C_{10} = C_0$

Receive nothing; just conclude $C_{10} = 2$.

Enumeration example (8)

From the decompressor's point of view:

$N = 8$

Length	Substrings			
1	6×0		2×1	
2	4×00	2×01	2×10	?$\times11$
...				

Knowing: $C_{10} + C_{11} = C_1$

Receive nothing; just conclude $C_{11} = 0$.

And so on...

Bidirectional predictions

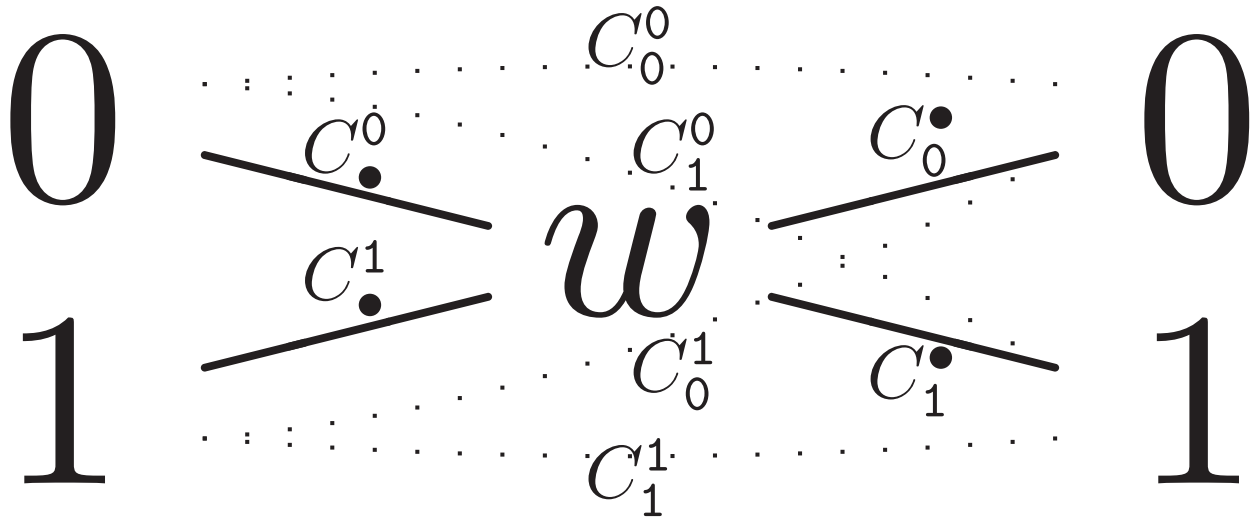
Natural equation: $C_{w\mathtt{0}} + C_{w\mathtt{1}} = C_w$.

Easier-to-forget equation: $C_{\mathtt{0}w} + C_{\mathtt{1}w} = C_w$.

In fact, $C_{\mathtt{0}w\mathtt{0}}$, $C_{\mathtt{0}w\mathtt{1}}$, $C_{\mathtt{1}w\mathtt{0}}$, $C_{\mathtt{1}w\mathtt{1}}$ are all related.

Butterfly (1)

The four extensions of w and the associated counters:



The meaning of the counters:

Counter	$C_\bullet^\bullet$	$C_\bullet^0$	$C_\bullet^1$	$C_0^\bullet$	$C_1^\bullet$	C_0^0	C_1^0	C_0^1	C_1^1
# occur. of	w	$0w$	$1w$	$w0$	$w1$	$0w0$	$0w1$	$1w0$	$1w1$

Butterfly (2)

Counter	$C_\bullet^\bullet$	$C_\bullet^0$	$C_\bullet^1$	$C_0^\bullet$	$C_1^\bullet$	C_0^0	C_1^0	C_0^1	C_1^1
# occur. of	w	$0w$	$1w$	$w0$	$w1$	$0w0$	$0w1$	$1w0$	$1w1$

Equations relating the counters together:

$$C_\bullet^0 + C_\bullet^1 \quad = \quad C_\bullet^\bullet \quad = \quad C_0^\bullet + C_1^\bullet$$

$$\begin{aligned} C_\bullet^0 &= C_0^0 + C_1^0 \\ C_\bullet^1 &= C_0^1 + C_1^1 \end{aligned} \qquad\qquad \begin{aligned} C_0^\bullet &= C_0^0 + C_0^1 \\ C_1^\bullet &= C_1^0 + C_1^1 \end{aligned}$$

Butterfly (3)

Each unknown counter has to be non-negative:

$$\begin{array}{rcl} C_0^0 & \geq & 0 \\ C_1^0 & \geq & 0 \\ C_0^1 & \geq & 0 \\ C_1^1 & \geq & 0 \end{array} \quad \Leftrightarrow \quad \begin{array}{rcl} C_0^0 & \geq & 0 \\ C_0^0 & \leq & C_\bullet^0 \\ C_0^0 & \leq & C_0^\bullet \\ C_0^0 & \geq & C_\bullet^0 - C_1^\bullet \end{array}$$

which results in the following bounds:

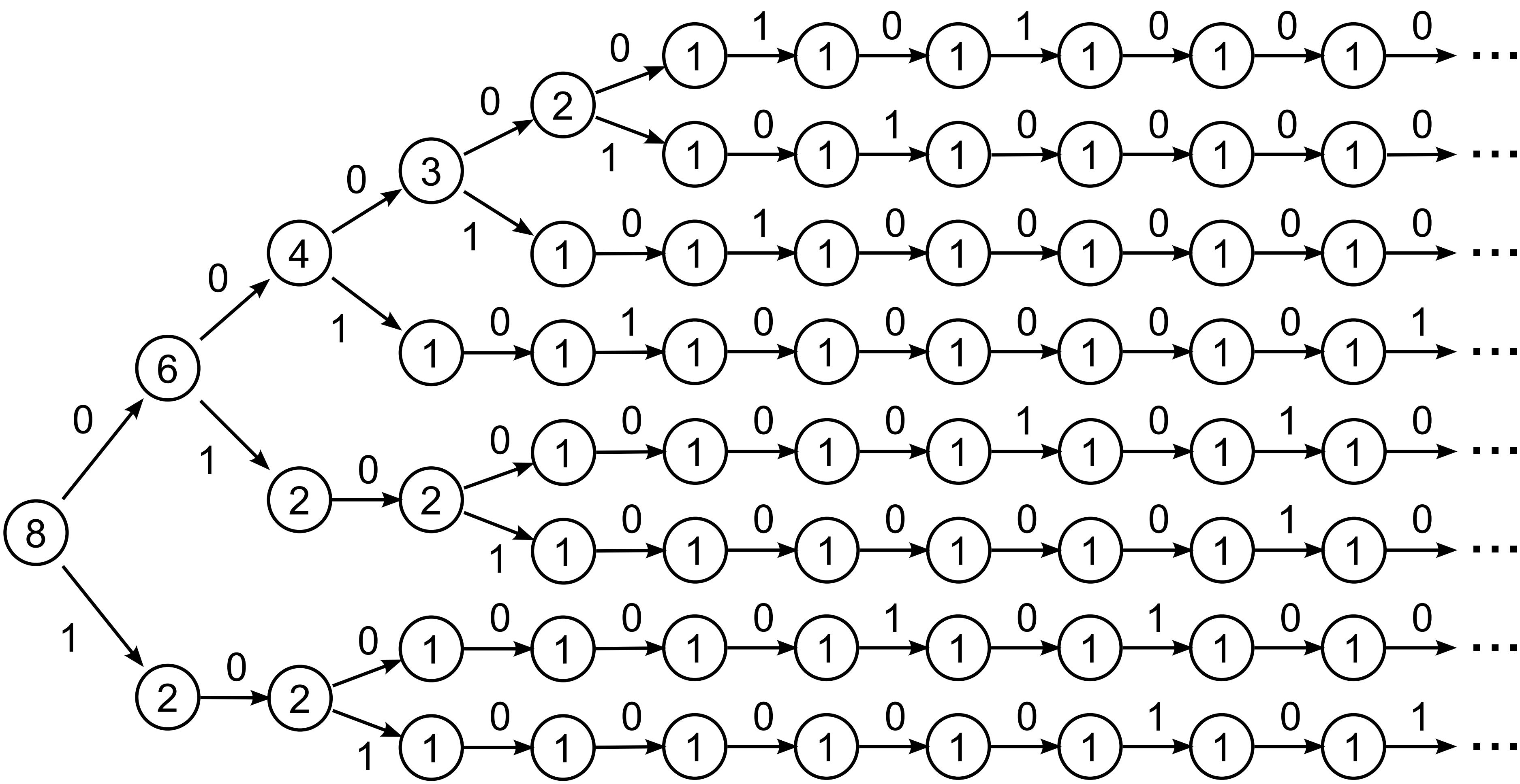
$$\max(0, C_\bullet^0 - C_1^\bullet) \quad \leq \quad C_0^0 \quad \leq \quad \min(C_\bullet^0, C_0^\bullet).$$

Reasons behind the actual compression

- The set of l-bit strings is a good summary for the set of $(l+1)$-bit strings
- Fewer counters on higher levels
- Butterflies take all the available local information into account
- A majority of the butterflies (almost 78%) are trivial
- For almost all butterflies ($> 99\%$), the min–max range is narrow (at most 23 possibilities)

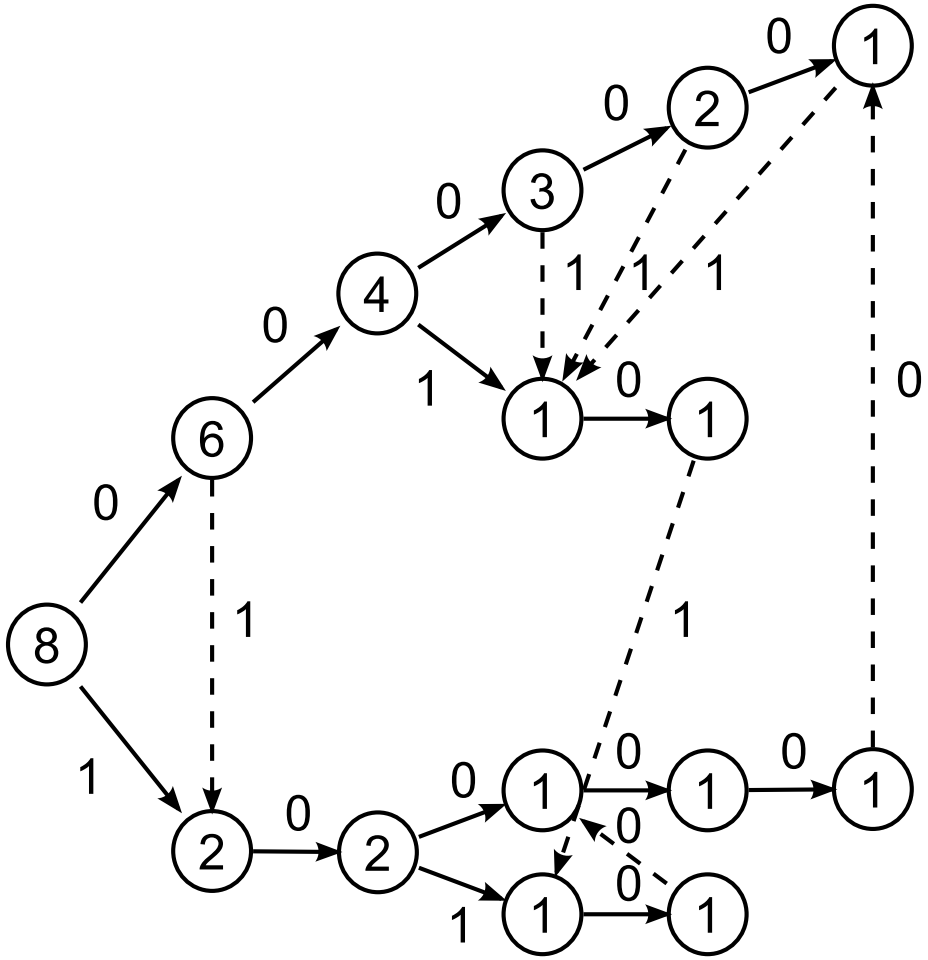
IST: infinite substring tree

The IST for $\mathbf{D} = \texttt{01000001}$.



CST: compact substring tree

The CST for $\mathbf{D} = 01000001$.



A CST has $2N - 1$ nodes, if $\mathbf{D}$ is non-repetitive. Conjecture.

Basic idea: more precisely

Compression of $\mathbf{D} \in \{0,1\}^+$ using substring enumeration:

Send N
Send C_0
For l **:=** 2 **to** N **do**
  For every core w in the CST such that $|w| = l - 2$ **do**
    Solve butterfly for $0w0$, $0w1$, $1w0$, and $1w1$

At most $2N - 1$ numbers to send.

Summary of the CSE technique

Steps performed by the compressor:

- Handle cases where $\mathbf{D}$ is repetitive
- Build suffix tree for $\mathbf{D}^2$
- Convert the suffix tree into the CST for $\mathbf{D}$
- Transmit N, C_0, and other necessary C_{w_1}, C_{w_2}, ..., using butterflies
- Send $\mathbf{D}$'s rank

Steps performed by the decompressor:

- Build the CST for $\mathbf{D}$ by receiving N, C_0, and other C_{w_1}, C_{w_2}, ...
- Recover $\mathbf{D}$ from its rank
- Handle cases where $\mathbf{D}$ is repetitive

Experimental results (1)

Techniques being compared:

- **Gzip**: `gzip` set at maximal compression
- **BWT**: the Burrows-Wheeler transform (from [2])
- **PPM**: PPM*C (from [2])
- $\overline{\textbf{Btf}}$: prototype, 32 kB blocks, flat predictions
- **Btf** : prototype, 32 kB blocks, adaptive predictions
- **BTF**: prototype, 1 MB blocks, adaptive predictions

[2] J. G. Cleary and W. J. Teahan. Unbounded length contexts for PPM. *The Computer Journal*, 40(2/3):67-75, 1997.

Experimental results (2)

File	Gzip	BWT	PPM	$\overline{\text{Btf}}$	Btf	BTF
bib	2.51	2.07	**1.91**	2.54	2.56	1.98
book1	3.25	2.49	2.40	3.14	3.06	**2.27**
book2	2.70	2.13	2.02	2.74	2.72	**1.98**
geo	5.34	**4.45**	4.83	6.03	5.52	5.35
news	3.06	2.59	**2.42**	3.33	3.32	2.52
obj1	**3.84**	3.98	4.00	5.10	4.46	4.46
obj2	2.63	2.64	**2.43**	3.03	3.02	2.71
paper1	2.79	2.55	**2.37**	2.79	2.80	2.54
paper2	2.89	2.51	**2.36**	2.77	2.77	2.41
paper3	3.11	—	—	2.95	2.96	**2.73**
paper4	3.33	—	—	**3.17**	3.20	3.20
paper5	3.34	—	—	**3.29**	3.33	3.33
paper6	2.77	—	—	2.75	2.76	**2.65**
pic	0.82	0.83	0.85	2.05	0.79	**0.77**
progc	2.68	2.58	**2.40**	2.76	2.77	2.60
progl	1.80	1.80	**1.67**	1.90	1.89	1.71
progp	1.81	1.79	**1.62**	1.99	1.96	1.78
trans	1.61	1.57	**1.45**	2.16	2.07	1.60

Links of CSE with other techniques

- With prediction by partial matching (PPM)
  $\Rightarrow$ knowing C_{w0} and C_{w1} makes order-$|w|$ predictions possible
  $\Rightarrow$ predicts order-$(|w|+1)$ models, not individual bits

- With anti-dictionaries
  $\Rightarrow$ if w is an anti-word, then $C_w = 0$

- With LZ77 and LZ78
  $\Rightarrow$ recurring words lead to highly reliable / certain predictions

- With the Burrows-Wheeler transform (BWT)
  $\Rightarrow$ block-based
  $\Rightarrow$ prediction contexts grow up to full block

Future work

- Better prediction method for C_0^0
- Demonstration that substring enumeration is universal (given an appropriate predictor for C_0^0) Conjecture
- Demonstration that the size of the CST is $2N - 1$ (to appear)
- Investigation on the penalty incurred for losing the phase
- Exploitation of the existence of a Hamiltonian circuit

Questions?

Web page:
`http://w3.ift.ulaval.ca/~dadub100/`

Work supported by the Natural Science and Engineering Research Council of Canada

Using Synchronization Bits to Boost Compression by Substring Enumeration

Danny Dubé

Université Laval
Quebec City, Quebec, Canada

ISITA'10 — Taichung, Taiwan — October 18, 2010

Introduction

Compression by Substring Enumeration (CSE) introduced during DCC'10.

- Lossless general-purpose data compression.
- Competitive.
- Works at the bit level.
- Weaker on binary data (as opposed to text-like data).

Hypothesis: CSE is unaware of the phase of the bits and has more difficulty to infer it when working on binary data.

Our approach: to insert synchronization bits in the data.

Plan of the presentation

- Basics of CSE
- CSE's weakness
- Synchronization
- Experiments
- Future work

Basics of CSE (1)

Compression of $\mathbf{D} \in \{0,1\}^+$ using substring enumeration, where $N = |\mathbf{D}|$:

For l **:=** 1 **to** N **do**
  For every distinct l-bit substring w of $\mathbf{D}$ **do**
    Send number of occurrences of w in $\mathbf{D}$

$O(N^2)$ numbers to send!

Can be implemented in $O(N)$ time and space.

Basics of CSE (2)

Example on $\mathbf{D} = 01000001$:

Length	Substrings							
1	6×0						2×1	
2	4×00				2×01		2×10	
3	3×000			1×001	2×010		1×100	1×101
4	2×0000		1×0001	1×0010	1×0100	1×0101	1×1000	1×1010
5	1×00000	1×00001	1×00010	1×00101	1×01000	1×01010	1×10000	1×10100
6	1×000001	1×000010	1×000101	1×001010	1×010000	1×010100	1×100000	1×101000
7	1×0000010	1×0000101	1×0001010	1×0010100	1×0100000	1×0101000	1×1000001	1×1010000
8	1×00000101	1×00001010	1×00010100	1×00101000	1×01000001	1×01010000	1×10000010	1×10100000

Basics of CSE (3)

The numbers of occurrences of

$$0w0,\ 0w1,\ 1w0,\ \text{and}\ 1w1$$

are predicted at once.

Prediction done knowing the numbers of occurrences of

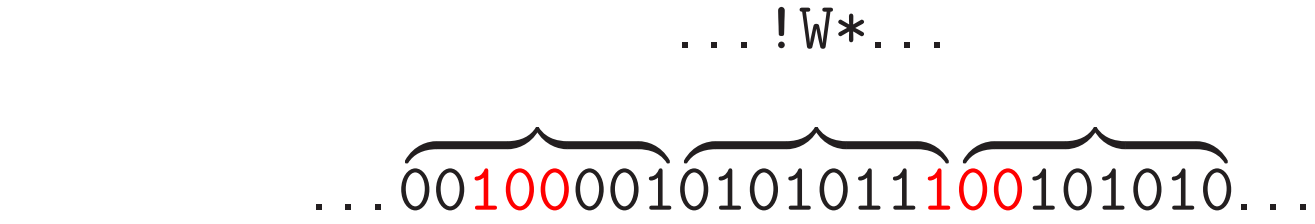
$$0w,\ 1w,\ w0,\ \text{and}\ w1.$$

Two important equations:

$$\begin{aligned} C_u &= C_{u0} + C_{u1} \\ C_v &= C_{0v} + C_{1v} \end{aligned}$$

Problems with phase unawareness

Example: compression of executable code



Bits on different phases (i.e. that have different significances) might have different statistics!

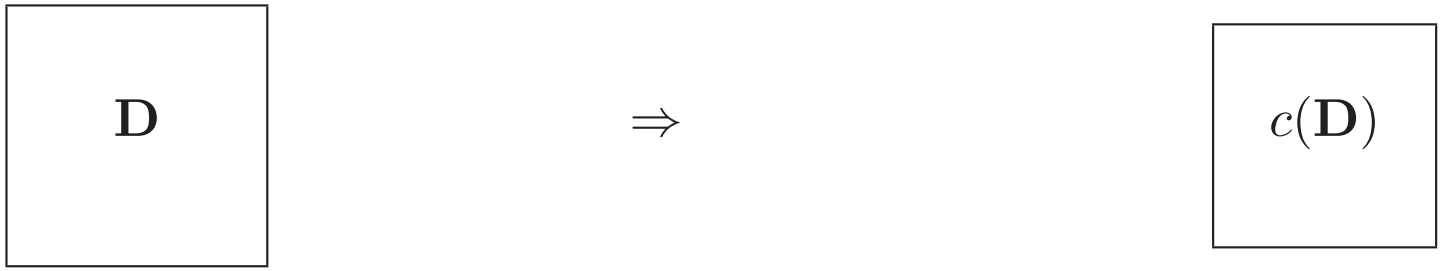
Fewer clues in binary data for CSE to infer the phase.

So suggests the hypothesis...

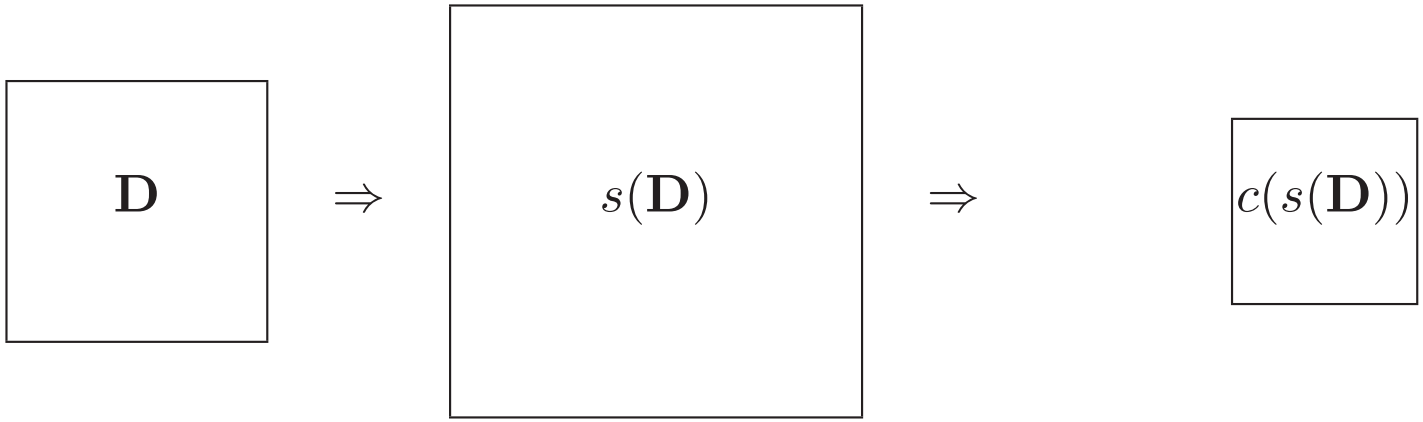
Synchronization (1)

Goal: inserting synchronization bits in the data.

Instead of:



we intend to try:



Synchronization (2)

We only use very simple synchronization schemes.

A synchronization scheme is characterized by its synchronization bit sequences: $w_1, w_2, \ldots, w_9 \in \{0,1\}^*$.

We perform insertion on a per-byte basis:

$$M(b_1\, b_2\, \ldots\, b_8) = w_1\, b_1\, w_2\, b_2\, \ldots\, w_8\, b_8\, w_9,$$

where $b_1\, \ldots\, b_8$ are the 8 bits of a byte.

A k-bit synchronization scheme is such that $|w_1\, \ldots\, w_9| = k$.

Synchronization (3)

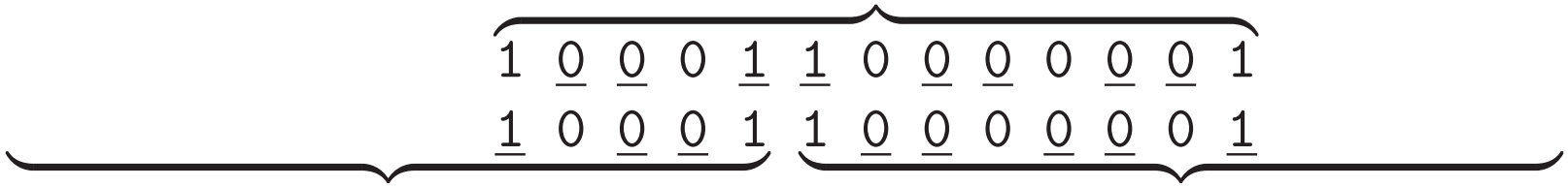
We say that a k-bit synchronization scheme provides *reliable* synchronization if two bits sequences that are:

- at least $(k + 8)$-bits long and
- on different phases

are necessarily different.

Synchronization (4)

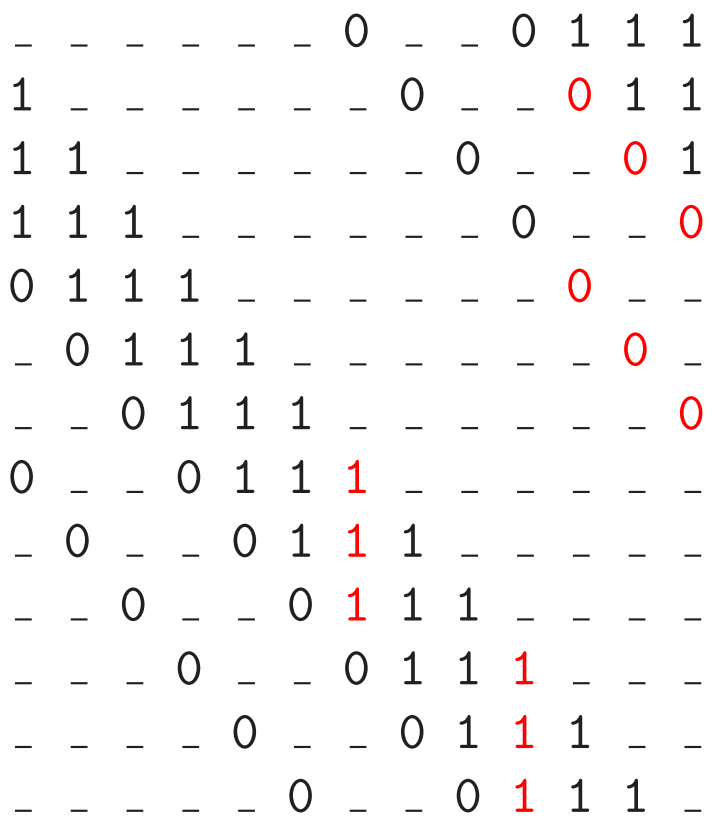
Example of a non-reliable 5-bit scheme,
where $w_1 = w_9 = 1$, $w_3 = w_5 = w_7 = 0$, and $w_2 = w_4 = w_6 = w_8 = \epsilon$:



Synchronization (5)

Example of a reliable 5-bit synchronization scheme:

It is such that
$w_1 = w_2 = w_3 = w_4 = w_5 = w_6 = w_8 = \epsilon$, $w_7 = 0$, and $w_9 = 0111$.



Experiments (1)

Parameters of the experiments:

- comparison of the Burrows-Wheeler transform, PPM*C, and an anti-dictionary technique with CSE with and without synchronization;
- CSE with 1- up to 5-bit synchronization schemes; the 5-bit scheme being the reliable one;
- 512 KB variant of CSE that learns predictions; and
- Calgary corpus files.

k	**Synchronization Scheme**
1	_ _ _ _ _ _ _ _ 0
2	_ _ _ _ _ _ _ _ 0 1
3	_ _ _ _ _ _ _ _ 0 1 1
4	_ _ _ _ _ _ _ _ 0 1 1 1
5	_ _ _ _ _ _ 0 _ _ 0 1 1 1

Experiments (2)

Raw results, in bits per character:

File	BWT	PPM	Anti	CSE	S-1	S-2	S-3	S-4	S-5
bib	2.07	1.91	2.56	1.98	1.95	1.92	1.92	1.91	**1.90**
book1	2.49	2.40	3.08	2.39	2.38	**2.37**	2.39	2.42	2.43
book2	2.13	**2.02**	2.81	2.07	2.06	2.06	2.06	2.05	2.04
geo	**4.45**	4.83	6.22	5.35	5.21	4.98	4.81	4.70	4.63
news	2.59	**2.42**	3.42	2.52	2.49	2.46	2.45	2.51	2.55
obj1	**3.98**	4.00	4.87	4.46	4.53	4.43	4.32	4.24	4.17
obj2	2.64	**2.43**	3.61	2.71	2.69	2.59	2.53	2.49	2.47
paper1	2.55	**2.37**	3.17	2.54	2.51	2.48	2.47	2.46	2.44
paper2	2.51	**2.36**	3.14	2.41	2.39	2.38	2.38	2.37	**2.36**
paper3	—	—	—	2.73	2.70	2.69	2.68	2.67	**2.65**
paper4	—	—	—	3.20	3.16	3.13	3.13	3.10	**3.07**
paper5	—	—	—	3.33	3.29	3.27	3.24	3.22	**3.19**
paper6	—	—	—	2.65	2.61	2.58	2.56	2.55	**2.52**
pic	0.83	0.85	1.09	**0.77**	0.84	0.83	0.83	0.84	0.83
progc	2.58	**2.40**	3.18	2.60	2.58	2.54	2.52	2.50	2.48
progl	1.80	1.67	2.24	1.71	1.70	1.69	1.68	1.67	**1.66**
progp	1.79	**1.62**	2.27	1.78	1.76	1.73	1.71	1.70	1.68
trans	1.57	**1.45**	1.94	1.60	1.58	1.53	1.52	1.50	1.48

Experiments (3)

Observations:

- The more numerous the synchronization bits, the better the compression (usually).
- Even non-reliable synchronization helps.
- No dramatic improvement when reliable synchronization is reached.
- Even text-like data is better compressed, but to a lesser extent than binary data.
- `pic` is already organized at the bit level; adding synchronization bits does not help.

Future work

- Using even more numerous synchronization bits in order to guarantee synchronization on substrings that are shorter than 13 bits.
- More sophisticated synchronization schemes.
- Use of colored bits; requires changing the inner workings of CSE.
- Working one bit plane at a time.

Questions?

Web page:
`http://w3.ift.ulaval.ca/~dadub100/`

Work supported by the Natural Science and Engineering Research Council of Canada

On the Use of Stronger Synchronization to Boost Compression by Substring Enumeration

(à venir)

Danny Dubé

Université Laval
Quebec City, Quebec, Canada

DCC'11 — Snowbird, Utah, USA — March 30, 2011

Stronger Reliability

If it is always possible to identify the phase by inspecting a finite number of bits starting at the given position, then we say that the synchronization scheme is *reliable*.

Otherwise, we say that the scheme is *unreliable*.

If a scheme is reliable and if it is always possible to identify the phase by inspecting at most n bits, then we say that the scheme is n-*reliable*.

Note that if a scheme is n-reliable, then it is also $(n+1)$-reliable.

If a scheme is reliable, then there exists an n such that the scheme is n-reliable.

Finally, if a scheme is not $(k+8)$-reliable, then it is unreliable.

Synchronization Schemes

n	k	Synchronization Scheme
—	0	_ _ _ _ _ _ _ _ _
—	1	_ _ _ _ _ _ _ _ _ 0
—	2	_ _ _ _ _ _ _ _ _ 0 1
—	3	_ _ _ _ _ _ _ _ _ 0 1 1
—	4	_ _ _ _ _ _ _ _ _ 0 1 1 1
13	5	_ _ _ _ _ _ 0 _ _ 0 1 1 1
12	8	_ _ _ 0 _ _ 1 0 0 _ _ 1 _ 1 1 0
11	8	_ _ _ 0 _ 0 _ _ 1 1 0 _ _ 1 1 0
10	10	_ _ _ _ 0 _ 0 1 1 _ _ _ 0 1 0 0 1 1
9	10	_ _ _ _ 0 0 0 1 1 _ _ _ _ 0 1 0 1 1
8	15	_ _ _ 0 0 0 1 0 _ _ _ 1 1 1 0 1 _ _ 1 1 0 0 1
7	20	1 1 0 1 _ 1 _ 1 1 0 0 _ 0 _ 0 1 0 0 _ 0 _ 1 1 0 0 _ 1 _

The Universality and Linearity of Compression by Substring Enumeration

(soumis)

Danny Dubé **Hidetoshi Yokoo**

Université Laval
Quebec City, Quebec, Canada

Gunma University
Kiryu, Gunma, Japan

Contributions

- Universalité de CSE,
  i.e. convergence de l'efficacité vers l'entropie pour des données provenant d'une source markovienne

- Complexité linéaire en temps et en espace