A Compact Representation of Nondeterministic (Suffix) Automata for the Bit-Parallel Approach

Domenico Cantone Simone Faro Emanuele Giaquinta

Department of Mathematics and Computer Science, University of Catania, Italy

CPM 2010

The problem

String Matching

Given strings $P \in \Sigma^m$ and $T \in \Sigma^n$, find all the occurrences of P in T.

Automata based solutions use

- the automaton $\mathcal{A}(P)$ for the language $\Sigma^* P$
- the automaton $\mathcal{S}(P)$ for the language $\{v \in \Sigma^* \mid \exists u \in \Sigma^*, P = uv\}$ of the suffixes of P

The problem

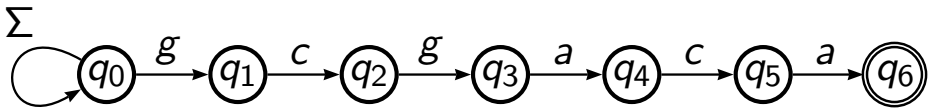


Figure: nondeterministic automaton for the string **gcgaca**

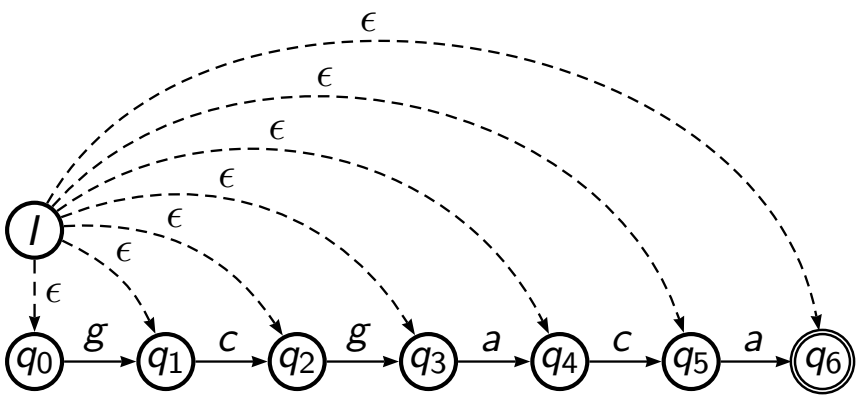


Figure: nondeterministic suffix automaton for the string **gcgaca**

The problem

Bit-Parallelism

Compact representation of nondeterministic $\mathcal{A}(P)$ and $\mathcal{S}(P)$ using bit-vectors.

- Automaton configuration encoded using $\lceil m/w \rceil$ computer words
- Transition function encoded using $|\Sigma|\lceil m/w \rceil$ computer words

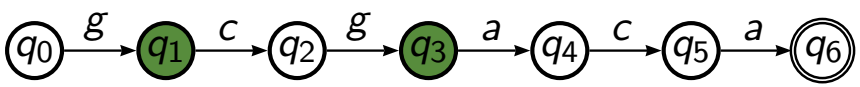
where w is the word size in bits.

Bit-Parallelism

- D vector of m bits, bit i set to 1 iff state q_{i+1} is active

Bit-Parallelism

- D vector of m bits, bit i set to 1 iff state q_{i+1} is active



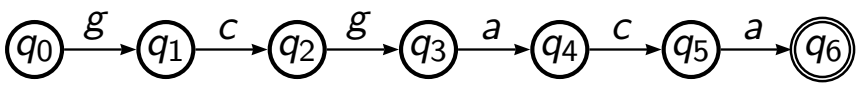
$$D = 000101 = \{q_1, q_3\}$$

Bit-Parallelism

- $B[c]$ vector of m bits, bit i set to 1 iff $\delta(q_i, c) = q_{i+1}$, $\forall c \in \Sigma$

Bit-Parallelism

- $B[c]$ vector of m bits, bit i set to 1 iff $\delta(q_i, c) = q_{i+1}$, $\forall c \in \Sigma$

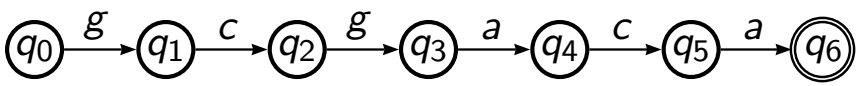


B=

σ	q_1/g	q_2/c	q_3/g	q_4/a	q_5/c	q_6/a
a	0	0	0	1	0	1
c	0	1	0	0	1	0
g	1	0	1	0	0	0

Bit-Parallelism

- $B[c]$ vector of m bits, bit i set to 1 iff $\delta(q_i, c) = q_{i+1}$, $\forall c \in \Sigma$



B=

σ	q_1/g	q_2/c	q_3/g	q_4/a	q_5/c	q_6/a
a	0	0	0	1	0	1
c	0	1	0	0	1	0
g	1	0	1	0	0	0

Let D_j be the configuration of $\mathcal{S}(P)$ at iteration j

$$
\begin{array}{lcll}
D_1 & \leftarrow & 1^m \,\&\, B[c] & Q \cap B[c] \\
D_j & \leftarrow & (D_{j-1} \ll 1) \,\&\, B[c] & \bigcup_{q_i \in D_{j-1}} \{q_{i+1}\} \cap B[c]
\end{array}
$$

Bit-Parallelism

Example

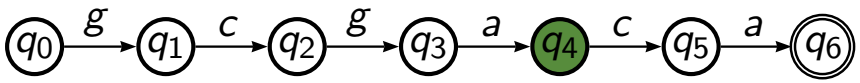
$$D = 000101 = \{q_1, q_3\}$$

Bit-Parallelism

Example

$$D = 000101 = \{q_1, q_3\}$$

$$\delta(D, a) = (000101 \ll 1)\ \&\ B[a] = 001010\ \&\ 101000 = 001000 = \{q_4\}$$



Bit-Parallelism

To verify if a configuration D contains the final state q_m, it suffices to check whether

$$(D \,\&\, 10^{m-1}) \neq 0$$

Bit-Parallelism

Problems

- $\lceil m/w \rceil$ overhead in time complexity if we build the automaton for the whole string
- maximum shift in BNDM-like algorithms bounded by w if we build the automaton for a prefix of P of length w

Can we do better?

1-factorization

A 1-**factorization** of size k of P is a sequence $\langle u_1, u_2, \ldots, u_k \rangle$ of nonempty substrings of P such that:

- $P = u_1 u_2 \ldots u_k$
- each factor u_j contains at most *one* occurrence of any of the characters in the alphabet Σ, for $j = 1, \ldots, k$

1-factorization

A 1-**factorization** of size k of P is a sequence $\langle u_1, u_2, \ldots, u_k \rangle$ of nonempty substrings of P such that:

- $P = u_1 u_2 \ldots u_k$
- each factor u_j contains at most *one* occurrence of any of the characters in the alphabet Σ, for $j = 1, \ldots, k$

Some facts:

- A 1-factorization is **minimal** if $\mathit{first}(u_{i+1})$ occurs in u_i, for $i = 1, \ldots, k-1$.

1-factorization

A 1-**factorization** of size k of P is a sequence $\langle u_1, u_2, \ldots, u_k \rangle$ of nonempty substrings of P such that:

- $P = u_1 u_2 \ldots u_k$
- each factor u_j contains at most *one* occurrence of any of the characters in the alphabet Σ, for $j = 1, \ldots, k$

Some facts:

- A 1-factorization is **minimal** if $first(u_{i+1})$ occurs in u_i, for $i = 1, \ldots, k-1$.
- $\left\lceil \frac{m}{|\Sigma|} \right\rceil \leq k \leq m$

1-factorization

Example

Minimal 1-factorization of *gcgaca*

$$< gc, gac, a >$$

1-factorization

A 1-factorization $\langle u_1, u_2, \ldots, u_k \rangle$ of a given pattern $P \in \Sigma^*$ induces naturally a partition $\{Q_1, \ldots, Q_k\}$ of the set $\{q_1, \ldots, q_m\}$ of states of $\mathcal{S}(P)$, where

$$Q_i =_{\text{Def}} \left\{ q_{\sum_{j=1}^{i-1} |u_j|+1}, \ldots, q_{\sum_{j=1}^{i} |u_j|} \right\}, \text{ for } i = 1, \ldots, k\,.$$

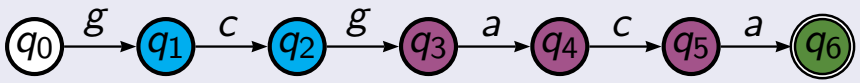
1-factorization

A 1-factorization $\langle u_1, u_2, \ldots, u_k \rangle$ of a given pattern $P \in \Sigma^*$ induces naturally a partition $\{Q_1, \ldots, Q_k\}$ of the set $\{q_1, \ldots, q_m\}$ of states of $\mathcal{S}(P)$, where

$$Q_i =_{\text{Def}} \left\{ q_{\sum_{j=1}^{i-1} |u_j|+1}, \ldots, q_{\sum_{j=1}^{i} |u_j|} \right\}, \text{ for } i = 1, \ldots, k\,.$$

Example

$$< gc, gac, a >$$



- $Q_1 = \{q_1, q_2\}$
- $Q_2 = \{q_3, q_4, q_5\}$
- $Q_3 = \{q_6\}$

1-factorization

By definition of 1-factorization, no two states in Q_i may have an incoming transition labeled by the same character.

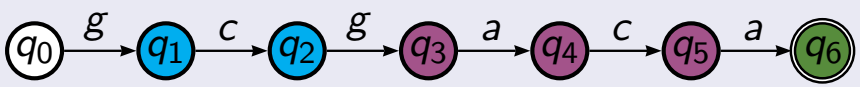
Let $q_{i,a}$ be the unique state in Q_i, if any, with an incoming transition labeled a.

1-factorization

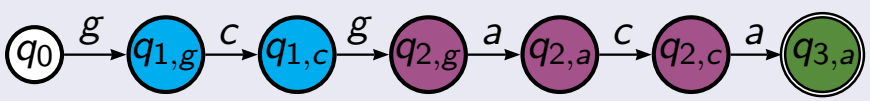
By definition of 1-factorization, no two states in Q_i may have an incoming transition labeled by the same character.

Let $q_{i,a}$ be the unique state in Q_i, if any, with an incoming transition labeled a.

Example



- $Q_1 = \{q_1, q_2\}$
- $Q_2 = \{q_3, q_4, q_5\}$
- $Q_3 = \{q_6\}$



Automaton configuration encoding

We encode a configuration of $\mathcal{S}(P)$ as the pair (D, a), where:

- D is a vector of k bits, where bit i is set to 1 iff Q_i contains an active state
- a is the last character read

Thus, if $i_1, i_2, \ldots, i_l$ are the indices of the bits set in D, the pair (D, a) represents the configuration of $\mathcal{S}(P)$

$$\{q_{i_1,a}, q_{i_2,a}, \ldots, q_{i_l,a}\}$$

Automaton configuration encoding

Example

- $Q_1 = \{q_1, q_2\}$
- $Q_2 = \{q_3, q_4, q_5\}$
- $Q_3 = \{q_6\}$

$$(D, a) = (011, g)$$

Transition function encoding

- (D, a) current configuration
- (D', c) new configuration after reading symbol c

$$(D, a) \xrightarrow{\mathcal{S}} (D', c)$$

For each $i = 1, \ldots, k$, bit i is set in D' if and only if either

(i) bit i is set in D, $last(u_i) \neq a$, and $q_{i,c} \in \delta(q_{i,a}, c)$, or

(ii) $i \geq 2$, bit $i-1$ is set in D, $last(u_{i-1}) = a$ and $q_{i,c} \in \delta(q_{i-1,a}, c)$

Transition function encoding

Given a 1-factorization $< u_1, u_2, \ldots, u_k >$, the **closure** of u_i, denoted by $\overline{u_i}$, is defined as:

- $u_i.first(u_{i+1})$, $1 \leq i < k$
- u_k, $i = k$

Transition function encoding

Given a 1-factorization $< u_1, u_2, \ldots, u_k >$, the **closure** of u_i, denoted by $\overline{u_i}$, is defined as:

- $u_i.first(u_{i+1})$, $1 \leq i < k$
- u_k, $i = k$

Example

$$< gc, gac, a >$$

- $u_1 = gc$
- $u_2 = gac$
- $u_3 = a$
- $\overline{u_1} = gcg$
- $\overline{u_2} = gaca$
- $\overline{u_3} = a$

Transition function encoding

We define a $|\Sigma| \times |\Sigma|$ matrix B and an array L of size $|\Sigma|$ of vectors of k bits:

- $\forall c_1, c_2 \in \Sigma$, the i-th bit of $B[c_1][c_2]$ is set to 1 **iff** the string $c_1 c_2$ is a factor of $\overline{u_i}$, or equivalently **iff** either
  - (i) $last(u_i) \neq c_1 \wedge q_{i,c_2} \in \delta(q_{i,c_1}, c_2)$, or
  - (ii) $i < k \wedge last(u_i) = c_1 \wedge q_{i+1,c_2} \in \delta(q_{i,c_1}, c_2)$

- $\forall c \in \Sigma$, the i-th bit of $L[c]$ is set to 1 **iff** $last(u_i) = c$

Transition function encoding

Example

$$< gc, gac, a >$$

- $u_1 = gc$
- $u_2 = gac$
- $u_3 = a$
- $\overline{u_1} = gcg$
- $\overline{u_2} = gaca$
- $\overline{u_3} = a$

B=

(σ, σ)	Q_3	Q_2	Q_1
(g, c)	0	0	1
(c, g)	0	0	1
(g, a)	0	1	0
(a, c)	0	1	0
(c, a)	0	1	0

L=

σ	Q_3	Q_2	Q_1
c	0	1	1
a	1	0	0

Transition encoding

$$(D, a) \xrightarrow{\mathcal{S}} (D', c)$$

Bitwise operations to perform a transition on symbol c from configuration (D, a):

$$\begin{array}{lcll} D & \leftarrow & D \,\&\, B[a][c] & D \cap \{1 \leq i \leq k \mid \delta(q_{i,a}, c) \neq \text{NIL}\} \\ H & \leftarrow & D \,\&\, L[a] & D \cap \{1 \leq i < k \mid \delta(q_{i,a}, c) = q_{i+1,c}\} \\ D' & \leftarrow & (D \,\&\, \sim H) \mid (H \ll 1) & D \setminus H \cup \{i+1 \mid i \in H\} \end{array}$$

Transition encoding

Example

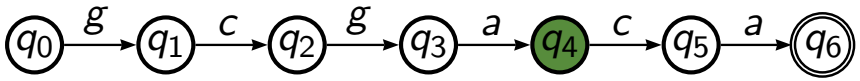
$$D = (011, g) = \{q_1, q_3\}$$

Transition encoding

Example

$$D = (011, g) = \{q_1, q_3\}$$

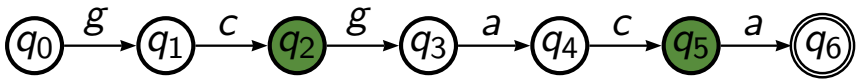
$$\begin{aligned} D &\leftarrow 011 \;\&\; B[g][a] = 011 \;\&\; 010 = 010 \\ H &\leftarrow 010 \;\&\; L[g] = 010 \;\&\; 000 = 000 \\ D &\leftarrow (010 \;\&\; \sim H) \mid (H \ll 1) = (010 \;\&\; 111) \mid 000 = 010 \end{aligned}$$



$$D = (010, a) = \{q_4\}$$

Transition encoding

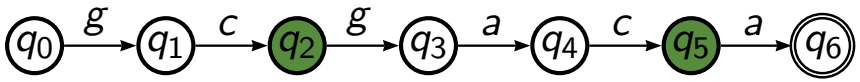
Example



$$D = (011, c) = \{q_2, q_5\}$$

Transition encoding

Example



$$D = (011, c) = \{q_2, q_5\}$$

$$\begin{aligned} D &\leftarrow 011 \;\&\; B[c][g] = 011 \;\&\; 001 = 001 \\ H &\leftarrow 001 \;\&\; L[c] = 001 \;\&\; 011 = 001 \\ D &\leftarrow (001 \;\&\; \sim H) \mid (H \ll 1) = (001 \;\&\; 110) \mid 010 = 010 \end{aligned}$$

$$D = (010, g) = \{q_3\}$$

Acceptance condition

To verify if a configuration (D, a) contains the final state $q_m = q_{k,last(u_k)}$ we must check that:

- bit k is set in D (one of the states in Q_k is active)
- $a = last(u_k)$

The test can be implemented with the following bitwise operations:

$$(D \;\&\; 10^{k-1} \;\&\; L[a]) \neq 0$$

Space complexity

- Bit-Parallelism encoding requires $|\Sigma|m$ bits
- 1-factorization encoding requires $(|\Sigma|^2 + |\Sigma|)k$ bits, where $\left\lceil \frac{m}{|\Sigma|} \right\rceil \leq k \leq m$ is the size of the 1-factorization

Trade-off: more rows for fewer columns!

The new encoding allows one to pack, in a machine word of w bits, strings longer than w on average.

Space complexity

	(A)		
m	$\|\Sigma\| = 4$	$\|\Sigma\| = 20$	$\|\Sigma\| = 63$
32	32	32	32
64	63	64	64
128	72	122	128
256	74	148	163
512	77	160	169
1024	79	168	173
1536	80	173	176
2048	80	174	178
4096	82	179	182

	(B)		
m	$\|\Sigma\| = 4$	$\|\Sigma\| = 20$	$\|\Sigma\| = 63$
32	15	8	6
64	29	14	12
128	59	31	26
256	119	60	50
512	236	116	102
1024	472	236	204
1536	705	355	304
2048	944	473	407
4096	1882	951	813

Table: (A) The average length of the longest substring of the pattern fitting in 32 bits. (B) The average size of the minimal 1-factorization of the pattern.

Experimental results

- Implementation in C, compiled with gcc 4, run on Intel Core 2 Duo 2GHz
- Alphabet sizes 4 (DNA), 20 (protein) and 63 (english)
- Set of 100 patterns of fixed length $m \in \{32, 64, 128, 256, 512, 1024, 1536, 2048, 4096\}$, randomly extracted from the text
- Comparison between the following algorithms[1]:
  - LBNDM
  - BNDM
  - BNDM*
  - F-BNDM
  - F-BNDM*

[1] * denotes the multi-word variant

Experimental results

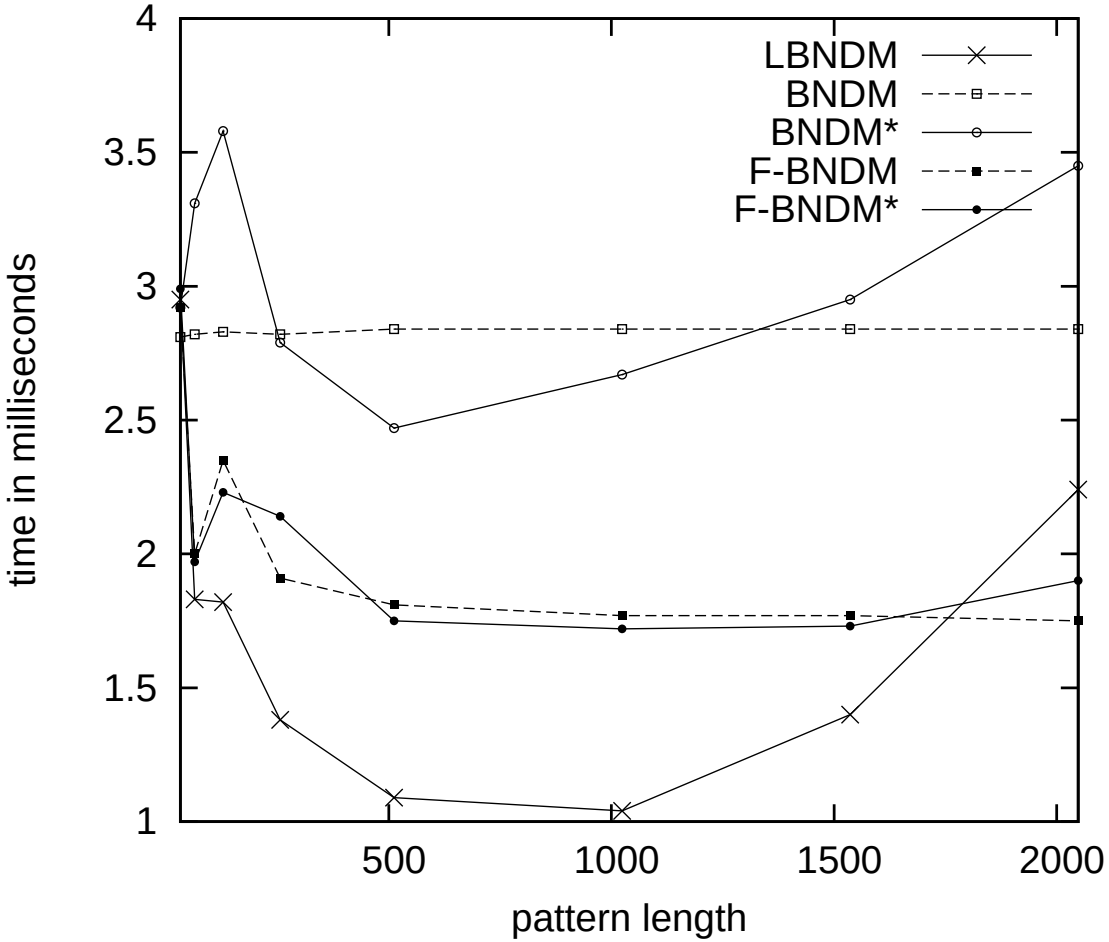


Figure: Experimental results on the King James version of the Bible ($|\Sigma| = 63$)

Experimental results

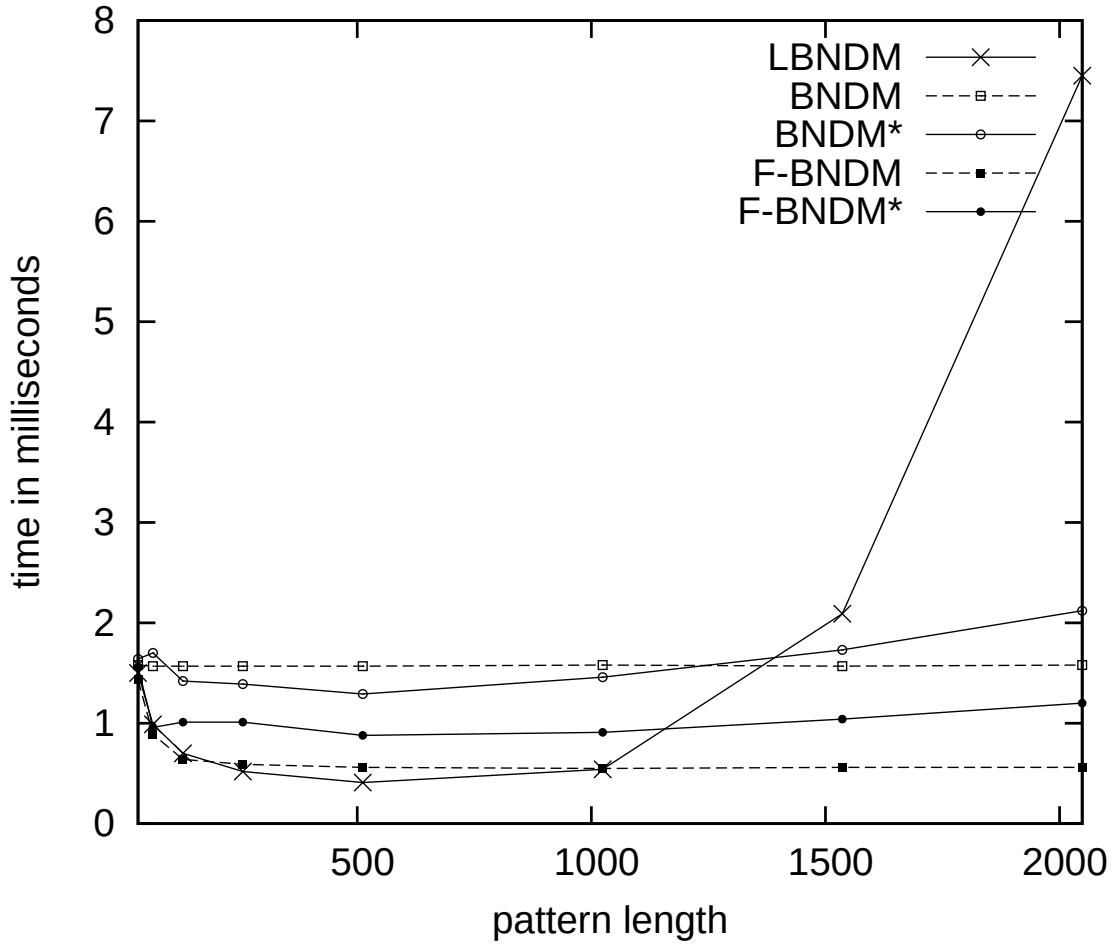


Figure: Experimental results on a protein sequence from the *Saccharomyces cerevisiae* genome ($|\Sigma| = 20$)

Experimental results

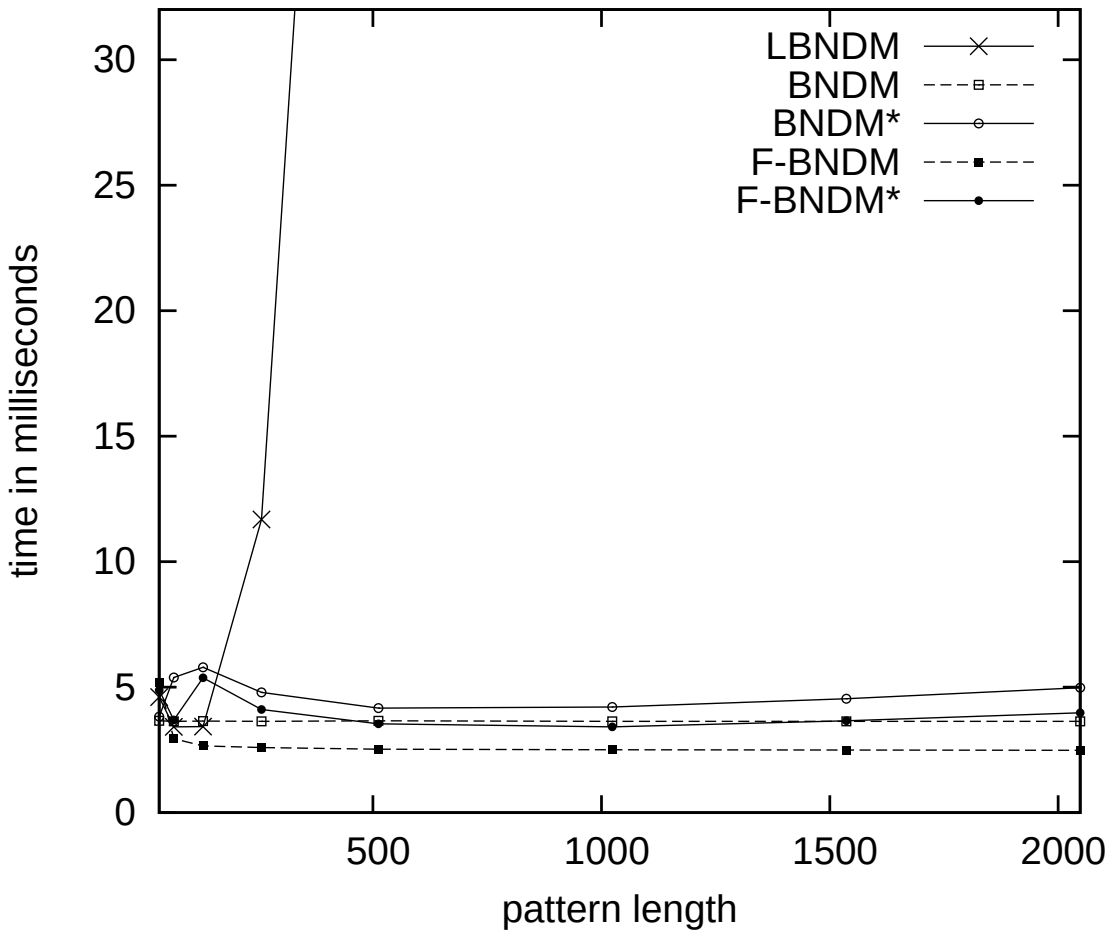


Figure: Experimental results on a genome sequence of *Escherichia coli* ($|\Sigma| = 4$)

Conclusions

- New technique, based on the bit-parallel approach, to encode (suffix) automata
- Fewer bits, compared to Bit-Parallelism, needed for the representation on average
- Longer shifts for BNDM-like algorithms for single word encodings

Conclusions

- New technique, based on the bit-parallel approach, to encode (suffix) automata
- Fewer bits, compared to Bit-Parallelism, needed for the representation on average
- Longer shifts for BNDM-like algorithms for single word encodings

Open problems

- Higher compactness using other kinds of factorizations
- Simpler encoding

$< thankyouf, ory, ourat, ten, tion >$