Compression of Biological Sequences by Greedy Off-line Textual Substitution

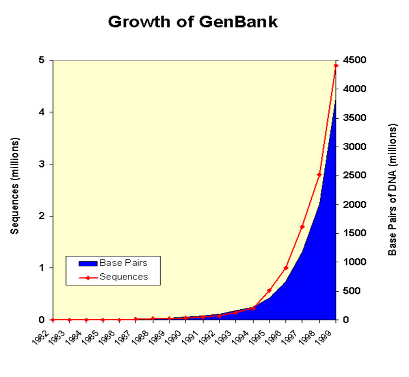
Alberto Apostolico *Stefano Lonardi*

Purdue University Università di Padova

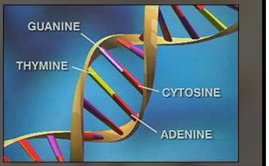
Genetic Databases

- Massive
- Growing exponentially

Example: GenBank contains approximately 4,654,000,000 bases in 5,355,000 sequence records as of December 1999



DNA Sequence Records



Composed by annotations (in English) and DNA bases (on the alphabet {A,C,G,T,U,M,R,W,S,Y,K,V,H,D,B,X,N})

```
>RTS2   RTS2 upstream sequence, from -200 to -1
TCTGTTATAGTACATATTATAGTACACCAATGTAAATCTGGTCCGGGTTACACAACACTT
TGTCCTGTACTTTGAAAACTGGAAAAACTCCGCTAGTTGAAATTAATATCAAATGGAAAA
GTCAGTATCATCATTCTTTTCTTGACAAGTCCTAAAAAGAGCGAAAACACAGGGTTGTTT
GATTGTAGAAAATCACAGCG
>MEK1   MEK1 upstream sequence, from -200 to -1
TTCCAATCATAAAGCATACCGTGGTYATTTAGCCGGGGAAAAGAAGAATGATGGCGGCTA
AATTTCGGCGGCTATTTCATTCATTCAAGTATAAAAGGGAGAGGTTTGACTAATTTTTTA
CTTGAGCTCCTTCTGGAGTGCTCTTGTACGTTTCAAATTTTATTAAGGACCAAATATACA
ACAGAAAGAAGAAGAGCGGA
>NDJ1   NDJ1 upstream sequence, from -200 to -1
ATAAAATCACTAAGACTAGCAACCACGTTTTGTTTTGTAGTTGAGAGTAATAGTTACAAA
TGGAAGATATATATCCGTTTCGTACTCAGTGACGTACCGGGCGTAGAAGTTGGGCGGCTA
TTTGACAGATATATCAAAAATATTGTCATGAACTATACCATATACAACTTAGGATAAAA
ATACAGGTAGAAAAACTATA
```

Problem

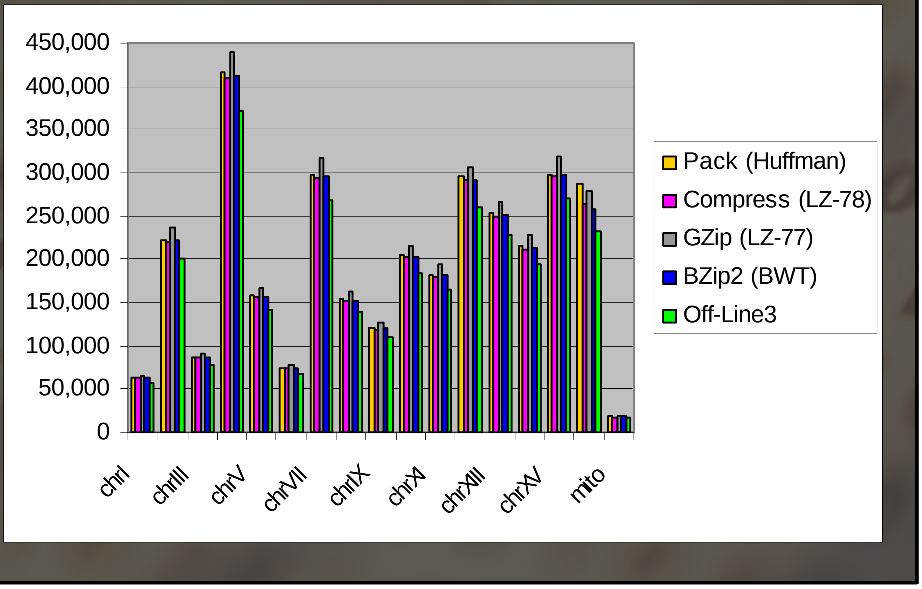
Textual compression of DNA data is *difficult,* i.e., “standard” methods do not seem to exploit the redundancies (if any) inherent to DNA sequences

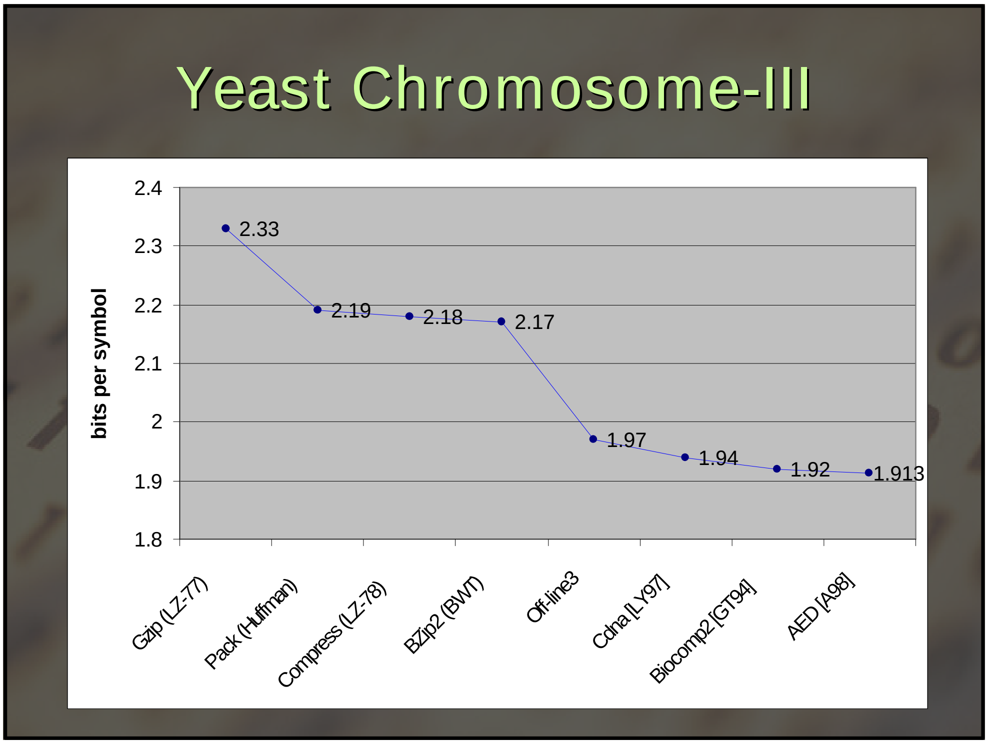
cfr. C.Nevill-Manning, I.H.Witten, “Protein is incompressible”, DCC99

Findings and Improvements

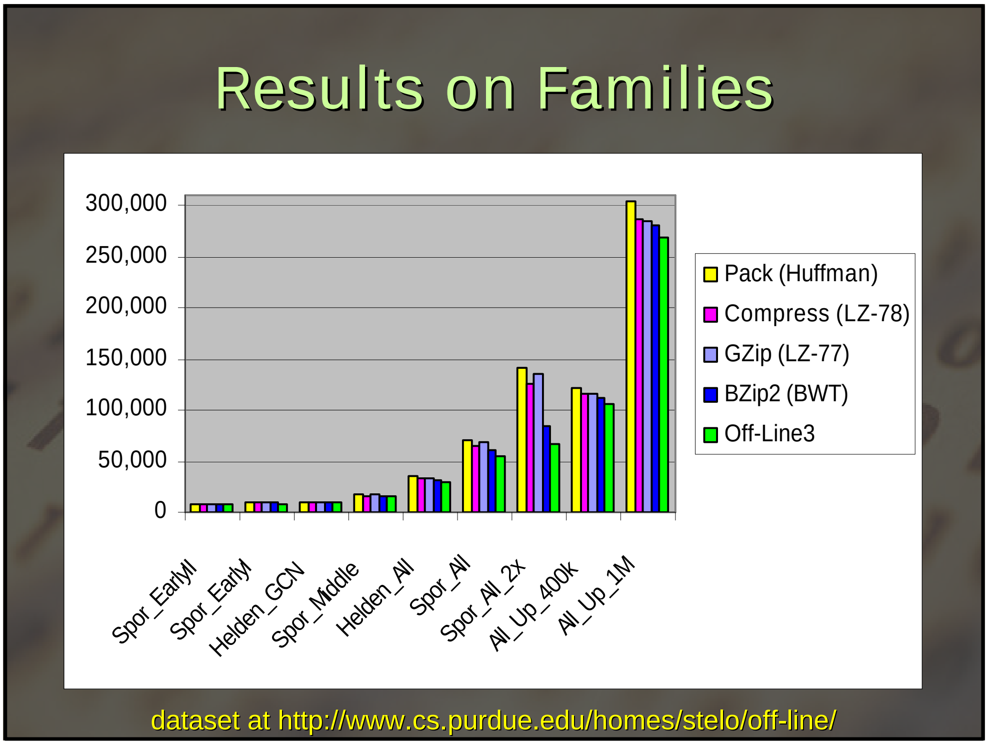
- A third scheme (**Off-line$_3$**) has been designed
- Compression time has been improved using a few “tuned” heuristics
- Compression performance on a single DNA sequences is substantially better than other generic textual compression methods
- Compression performance approaches the methods specifically designed for DNA sequences
- The best performance is in the compression of *families* of DNA sequences

Yeast Chromosomes





Yeast Chromosome-III



Results on Families

dataset at http://www.cs.purdue.edu/homes/stelo/off-line/

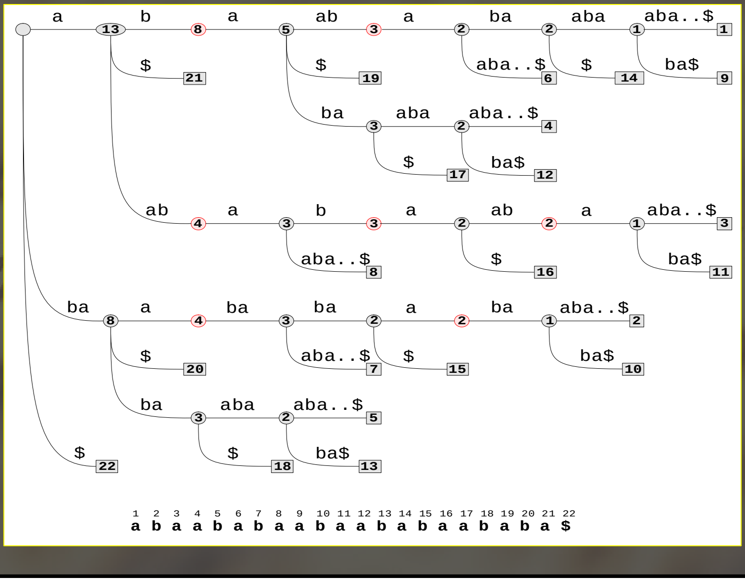
Overall Structure of Off-line

```
Off-line (string $x$)
repeat
    ▪ build an index $T$ of the substrings $w$ of the text $x$,
      and collect $f_w$ (count of non-overlapping occurrences)
    ▪ choose $Q$ substrings $s_1,\dots,s_Q$ in $T$ which maximize the
      gain function $G$
    ▪ substitute the occurrences of $s_1,\dots,s_Q$ in $x$ with
      pointers
until no further compression of $x$ can be obtained
```

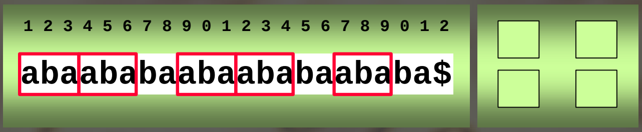
Data Structures

- index *T*: min. augmented suffix tree
  - construction $O(n \log^2(n))$
  - annotation with the count of non-overlapping occurrences $O(n)$
- text *x* stored in a balanced tree of text fragments
  - frequent deletions and string searches

Min. Augmented Suffix Tree

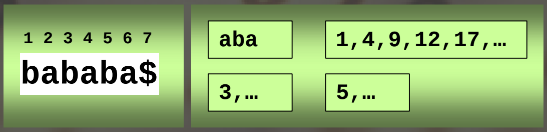


Off-line$_1$


1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2

abaababaabaababaababa$

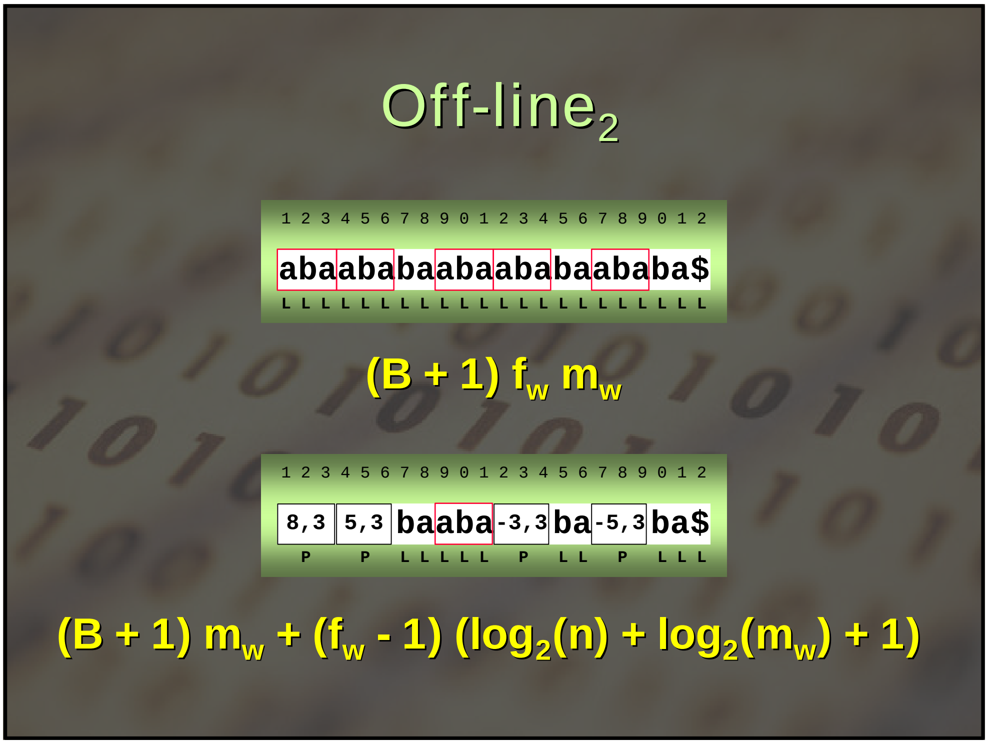
$$B\ f_w\ m_w$$


1 2 3 4 5 6 7

bababa$

aba	1,4,9,12,17,…
3,…	5,…

$$B\ m_w + \log_2(m_w) + \log_2(f_w) + f_w \log_2(n)$$



Off-line$_2$

1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2

abaababaabaababaababa$

L L L L L L L L L L L L L L L L L L L L L L

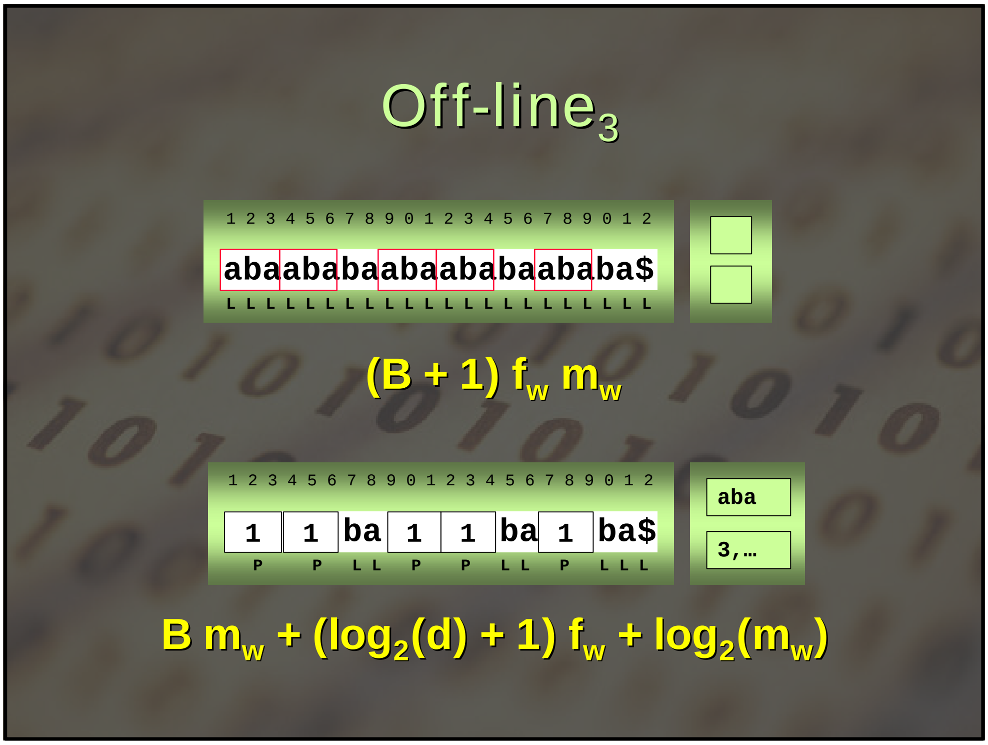
$(B + 1)\ f_w\ m_w$

1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2

8,3 5,3 baaba -3,3 ba -5,3 ba$

P P L L L L L P L L P L L L

$(B + 1)\ m_w + (f_w - 1)\ (\log_2(n) + \log_2(m_w) + 1)$



Off-line$_3$

1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2

abaababaabaababaababa$

L L L L L L L L L L L L L L L L L L L L L L

$(B + 1)\ f_w\ m_w$

1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2

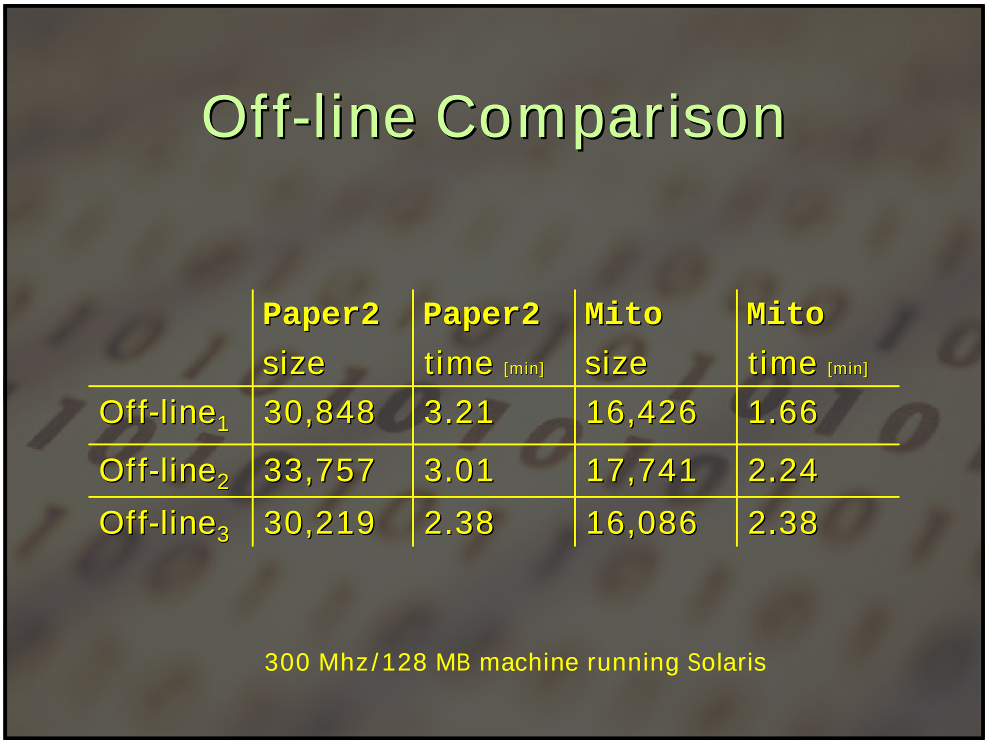
1 1 ba 1 1 ba 1 ba$

P P L L P P L L P L L L

aba

3,…

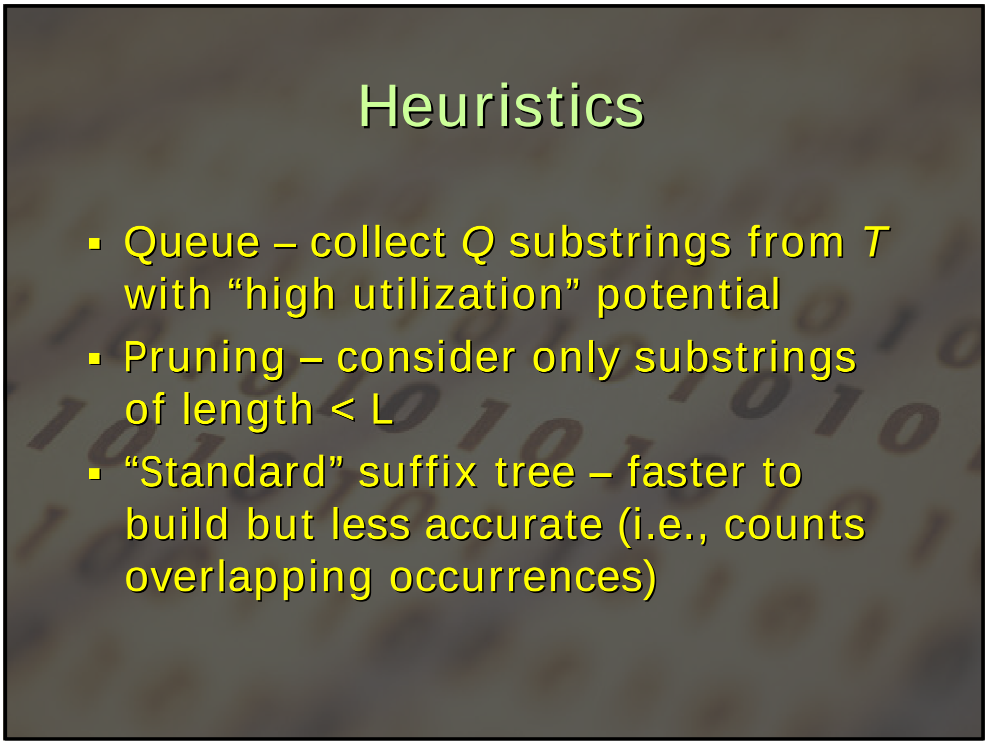
$B\ m_w + (\log_2(d) + 1)\ f_w + \log_2(m_w)$



Off-line Comparison

	Paper2	**Paper2**	**Mito**	**Mito**
	size	time [min]	size	time [min]
Off-line$_1$	30,848	3.21	16,426	1.66
Off-line$_2$	33,757	3.01	17,741	2.24
Off-line$_3$	30,219	2.38	16,086	2.38

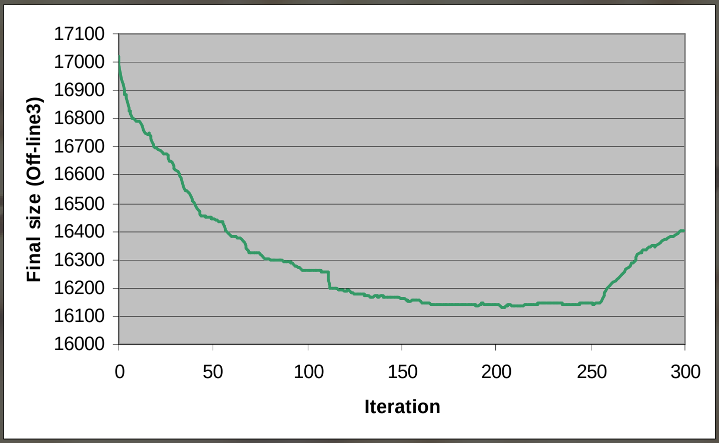
300 Mhz/128 MB machine running Solaris



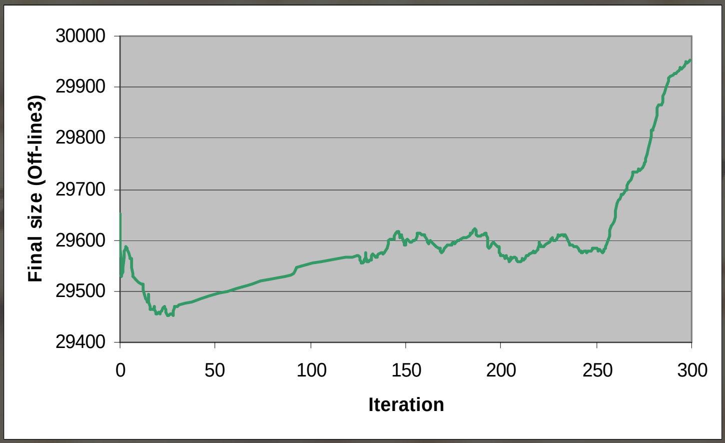
Heuristics

- Queue – collect *Q* substrings from *T* with “high utilization” potential
- Pruning – consider only substrings of length < L
- “Standard” suffix tree – faster to build but less accurate (i.e., counts overlapping occurrences)

Size vs. Iterations (**mito**)



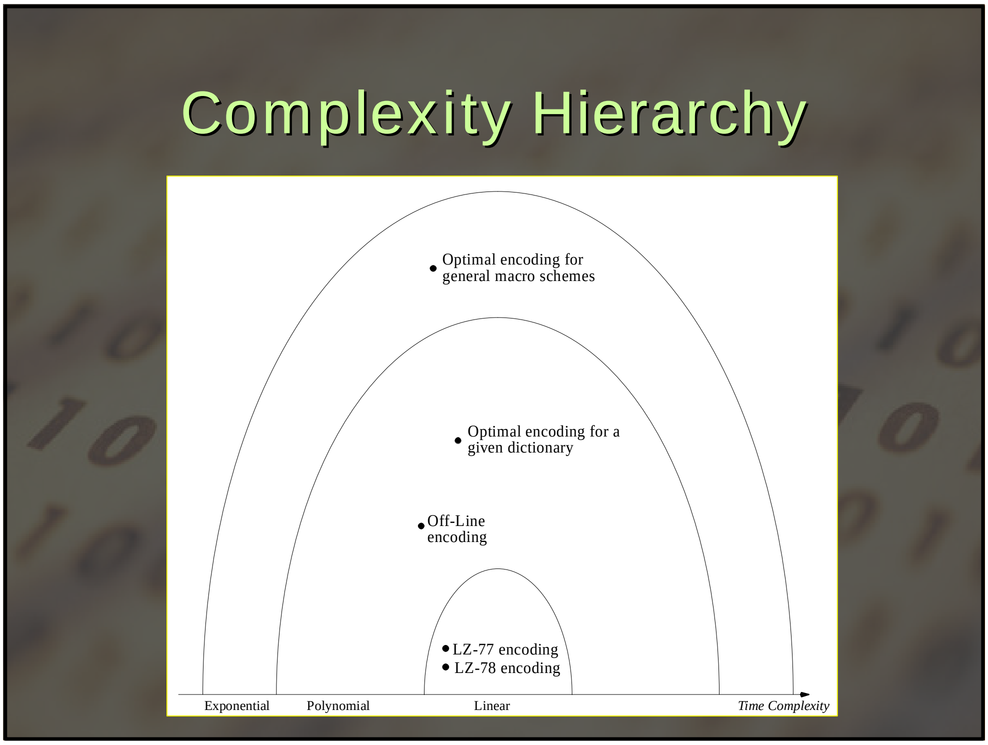
Size vs. Iterations (**paper2**)



Final Remarks

- **Off-line** appears to be a solid first step to tackle the problem of compression of genetic sequences
- *Next:* specialize **Off-line** for DNA with “biological knowledge” (e.g., palindromic/approximate occurrences)

Dictionary



Complexity Hierarchy

Optimal encoding for general macro schemes

Optimal encoding for a given dictionary

Off-Line encoding

LZ-77 encoding

LZ-78 encoding