DNA Compression Challenge Revisited: a Dynamic Programming Approach

Behshad Behzadi and Fabrice Le Fessant

LIX, Ecole Polytechnique, Paris, FRANCE

June 21 2005

Outline

1. DNA Compression Challenge
2. Tools and Methods
3. DNA Compression Algorithms
4. DNAPack
5. Results and Conclusion

Contents

1. DNA Compression Challenge
2. Tools and Methods
3. DNA Compression Algorithms
4. DNAPack
5. Results and Conclusion

DNA Compression Challenge

- DNA is a sequence of four bases.
- 2 bits per base is enough to encode DNA.
- We are only interested in lossless compression algorithms.

- Standard algorithms cannot compress DNA sequences !!!
  - HEHCMVCG: 229354 bases -> 57338 bytes without compression
  - With `gzip`: 66741 bytes
  - With `bzip2`: 62169 bytes

Motivation

- using less memory to store DNAs :)
- defining compression distance for making phylo. trees
- just a computer science challenge to compress better

Compression Distance

- $\frac{Comp(ST)+Comp(TS)}{Comp(S)+Comp(T)} - 1$
- phylogenetical trees on mithocondrial DNAs, ...

DNA Sequences Properties

- Existence of repeats in DNA sequences.
- Approximate repeats
- Complementary palindromes
- Local non-uniform frequencies of bases.

Contents

1 DNA Compression Challenge

2 Tools and Methods

3 DNA Compression Algorithms

4 DNAPack

5 Results and Conclusion

Encodings of Text

- Fix number of bits per symbol
- Huffman Encoding
- Adaptative Huffman Encoding
- Arithmetic Coding
- Adaptative Arithmetic Coding
- Context Tree Weighted method

Encoding of Numbers

- Fix number of bits per Number (bounded numbers only).
- Self Delimited Encodings:
  - Fibonacci encoding of the numbers.
  - k-shifted Fibonacci encoding:
    - $bin(n)$ if $n > 2^k$.
    - $0^k + fibo(n - (2^k - 1))$

	1	2	3	4	8	18
Fibonacci	11	011	0011	1011	000011	0001011
1-shifted Fibonacci	1	011	0011	00011	001011	01010011
3-shifted Fibonacci	001	010	011	100	00011	00001011

Contents

1 DNA Compression Challenge

2 Tools and Methods

3 DNA Compression Algorithms

4 DNAPack

5 Results and Conclusion

Previous Algorithms

- BioCompress (BioCompress-2)
- Cfact
- GenCompress-1 (GenCompress-2)
- CTW+LZ
- DNACompress
- DNASequitur
- . . .

BioCompress (Grumbach and Tahi 1994)

- Exact direct and reverse complementary repeats.
- At each step, the longest factor beginning at the current position which matches with a factor starting before is chosen.
- If there is no benefit the copy is encoded by two bits per base.
- BioCompress-2 uses arithmetic coding of order 2

Cfact (Rivals et al. 1996)

- Looks for longest exact matching repeat.
- Two passes (gain is guarranteed).
- Uses a suffix-tree for finding the longest repeat.
- 2 bits per base for non-repeat parts.

GenCompress (Chen et al. 1999)

- Approximate repeats are considered.
- At each step, looks for the *optimal prefix* (gain function) of the not yet encoded part (suffix) of the DNA sequence.
- No gain in using optimal prefix $\Rightarrow$ a letter is added to the buffer.
- Hamming distance (v1) and edit distance (v2) for approximate repeats.

CTW+LZ (Matsumoto et al. 2000)

- Combinaison of GenCompress and CTW (in place of arithmetic-2 coding).
- CTW: Context Tree Weighting method.
- Local heuristics for resolving the greedy selection problem.
- Bad execution time.

DNACompress (Chen et al. 2002)

- Uses PatternHunter as preprocessing.
- Found repeats are sorted in decreasing order of size (or gain function).
- While list not empty
  - Select the first repeat in the list
  - Remove overlapping repeats from the list
- Good execution time.

DNASequitur (Cherniavsky and Ladner 2004)

- Grammar based compression
- Sequitur (Nevill-Manning and Witten 1997)
  - *Digram Uniqueness*: no pair of adjacent symbols appears more than once in the grammar.
  - *Rule Utility*: each rule is used at least twice (except for the start rule).
- DNAsequitur: modified version of Sequitur adapted for DNA sequences.
  - The reverse complement of a string s is denoted by s'.

Common Components of most of DNA compression Algorithms

- Finding the candidate repeat segments.
- Considering approximate repeats.
- Selecting the best subset of compatible repeats.
- Encoding of the repeat segments.
- Encoding of the non-repeat segments.

Contents

1 DNA Compression Challenge

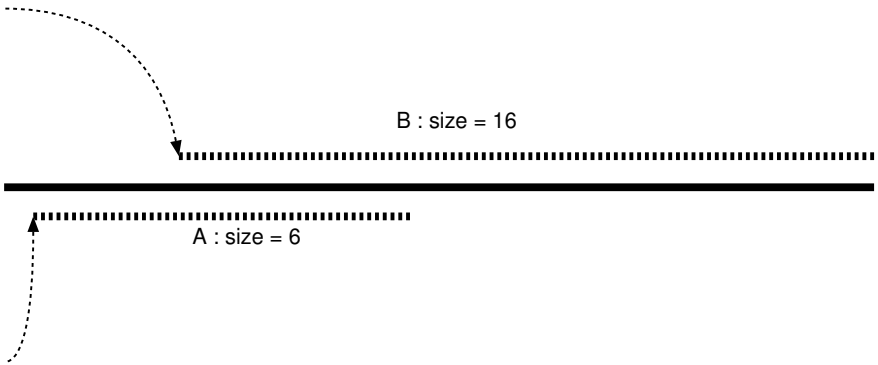
2 Tools and Methods

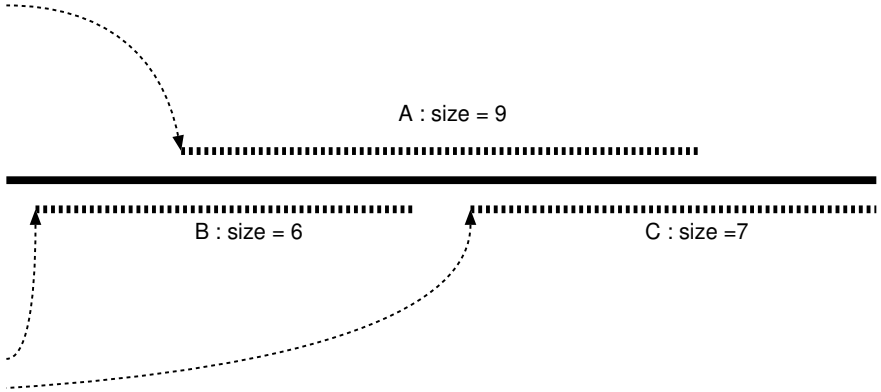
3 DNA Compression Algorithms

4 DNAPack

5 Results and Conclusion

Why not Greedy?





- Greedy approach of GenCompress (in left) does not produce the optimal.
- Greedy approach of DNACompress (in right) does not produce the optimal.

DNAPack

- Dynamic Programming instead of Greedy Algorithms
- Heurestics make the Dynamic Programming appliable on large sequences.

DNAPack: Dynamic programming

- *BestComp*[*i*]: minimum number of bits needed to encode $T[1..i]$.
- *BestCopy*[*i*]: minimum number of bits needed to encode $T[1..i]$ such that the last segment is encoded as a repeat segment.

Dynamic Programming Algorithm

$CopyCost(j,i,k)$: min. # bits to describe $t[j+1..j+k] \longrightarrow t[i-k+1..i]$
$MinCost(j+1,i)$: min. # bits to encode $t[j+1..i]$ as a non-repeat seg.

DNAPack
Initialization: $BestComp[0] = 0$

Recurrence:

$$\forall i > 0 \qquad BestComp[i] = \min \begin{cases} BestComp[j] + CopyCost(j,i,k) & \forall k \ \forall\, 0 < j < i \\ BestComp[j] + PalinCopyCost(j,i,k) & \forall k \ \forall\, 0 < j < i \\ BestCopy[j] + MinCost(j+1,i) & \forall\, 0 < j < i \end{cases}$$

Reducing the Execution time (1)

- Repeats with common starting substrings of size l (*seeds*)
- Hash-table on the *seeds*
- by increasing the size of *l* one can reduce the execution time.

Reducing Execution time (2)

- $MinCost(j+1, i)$ needs to be computed for values of j which $BestComp[j]$ is optimized by a repeat segment.
- Maintaining a list of positions satisfying $BestComp[j] = BestCopy[j]$
- It is possible to only maintain the positions of this list satisfying $RefBestCopy[j] \neq RefBestCopy[j-1]$

Reducing Execution time (3)

- in the case of repeats, if $s[i-1] = s[j-1]$ then there is no need to have a loop on k.
  $BC[j-1] + CopyCost(j-1, i, k+1) \leq BC[j] + CopyCost(j, i, k)$

Structure of the Compressed File

- HEADER
- CODE
- BASES

HEADER Region

- The type of compression used for sequences of bases
  - Arith-2
  - CTW
  - 2-bits
- The number of segments in the CODE part.
- The minimum size l of the first copy operation in a repeat.
- For each base, the most frequently observed substitution in a repeat.

CODE Region

- The CODE region consists of two types of segments
  - repeats
  - non-repeats
- There are no two consecutive non-repeat segments.
  - `-` empty code for first segment of DNA (non-repeat)
  - `0` for a non-repeat seg. after a repeat seg.
  - `1` for a repeat seg. after a repeat seg.
  - `-` empty code for a repeat seg. after a non-repeat seg.

Encoding of Non-Repeats

A non-repeat segment just contains the following information:

- The number of bases of the segment that will be found in the BASES region.

Encoding of Repeats(1)

A repeat segment must contain the following information:

- The type of the repeat (direct or comp.-palin.)
- The offset of the origin of the repeat.
- The sequence of operations which transforms the reference string into the actual repeat.

Encoding of Repeats(2)

Operation set = { Replace(x), Copy(k), Finish }

Operation opcodes:

- `0` for Finish after a Copy
- `1` for a Replace after a Copy
- `0` for a Copy after a Replace
- `1` for a Replace after a Replace

Encoding of Repeats(3)

Operation arguments:

- Replace(x):
  - `0` if x is the most probable substitution
  - `10` and `11` for the two other bases.
- Copy(k):
  - Fibonacci-encoding of k.
  - For the first copy, $fibo(k - l)$.

Implementation

We implemented DNAPack:

- 2500 lines of Objective-Caml for encoding/decoding
- Use external implementation of CTW by Koninklijke KPN NV (Netherlands)

Contents

1 DNA Compression Challenge

2 Tools and Methods

3 DNA Compression Algorithms

4 DNAPack

5 Results and Conclusion

Comparison of Results

sequence	length	BioComp-2	GenComp	CTW-LZ	DNAComp	DNAPack
CHMPXX	121024	1.6848	1.6730	1.6690	1.6716	**1.6602**
CHNTXX	155844	1.6172	1.6146	1.6120	1.6127	**1.6103**
HEHCMVCG	229354	1.8480	1.8470	1.8414	1.8492	**1.8346**
HUMDYSTROP	33770	1.9262	1.9231	1.9175	1.9116	**1.9088**
HUMGHCSA	66495	1.3074	1.0969	1.0972	**1.0272**	1.039
HUMHBB	73308	1.8800	1.8204	1.8082	1.7897	**1.7771**
HUMHDABCD	58864	1.8770	1.8192	1.8218	1.7951	**1.7394**
HUMHPRTB	56737	1.9066	1.8466	1.8433	1.8165	**1.7886**
MPOMTCG	186609	1.9378	1.9058	1.9000	**1.8920**	1.8932
PANMTPACGA	100314	1.8752	1.8624	1.8555	1.8556	**1.8535**
VACCG	191737	1.7614	1.7614	1.7616	**1.7580**	1.7583
Average	—	1.7837	1.7428	1.7389	1.7254	**1.7148**

Results

- Our contributions:
  - An efficient tool to compress DNA sequences using dynamic programming and approximate repeats.
  - A clear description of our encodings.
- Future work:
  - Adapt DNAPack to larger sequences of genes
  - Use DNAPack to compare sequences of genes