DNA Compression Algorithms

Behshad Behzadi & Fabrice Le Fessant
Ecole Polytechnique - France

DNA compression Challenge

- DNA is a sequence of four base.
- 2 bits per base is enough for coding a DNA.
- Standard algorithms cannot compress DNA sequences!!!
- We are interested in lossless compression algorithms.

DNA sequences properties

- Existance of the repeats in the DNA sequences.
- Approximate repeats
- complementary palindromes

Text Encodings

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

Encoding of the numbers

- Fix number of bits per Number.
- Self Delimited Encoding.
- Fibonacci encoding of the numbers.
- Shifted Fibonacci encoding

Existing algorithms

- BioCompress (BioCompress-2)
- Cfact
- GenCompress-1 (GenCompress-2)
- CTW+LZ
- DNACompress
- DNAC
- DNASequitor
- DNAPack

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 coded 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 garanteed)
- Uses a suffix-tree for finding the longest repeat.

GenCompress (Chen et al. 1999)

- Approximate repeats are considered.
- One pass algorithm.
- At each step, looks for the *optimal prefix* of the not yet encoded part (suffix) of the DNA sequence.
- no gain with optimal $\rightarrow$ a letter is added to the buffer.
- hamming distance and edit distance for approximated repeats.

CTW+LZ (Matsumoto et al. 2000)

- Similar to GenCompress but using CTW in place of arith-2
- CTW : context tree weighting method

DNACompress (Chen et al. 2002)

- Uses another sofware called PatternHunter as preprocessing.
- After finding repeats they are being chosen in the decreasing order of size (or gain function).
- only the *compatible repeats* are chosen.
- low execution time.

DNASequitor (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'.

DNAPack (Behzadi and Le Fessant 2005)

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

Dynamic Programming

Bellman's Principle of Optimality :
An optimal policy has the property that whatever the initial state and the initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions

Rutherford Aris : *IF you don't do the best with what you have happened to have got, you will never do the best with what you should have had.*

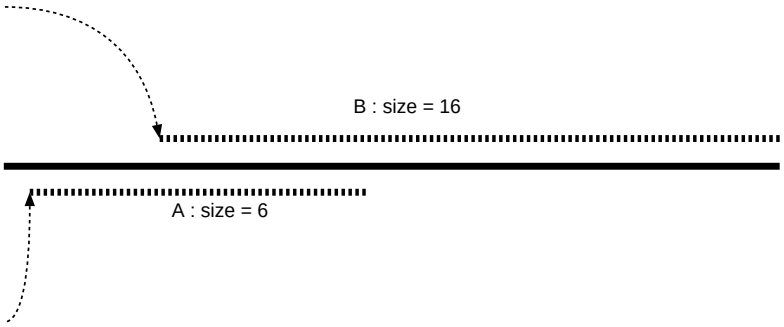
Dynamic Programming 2

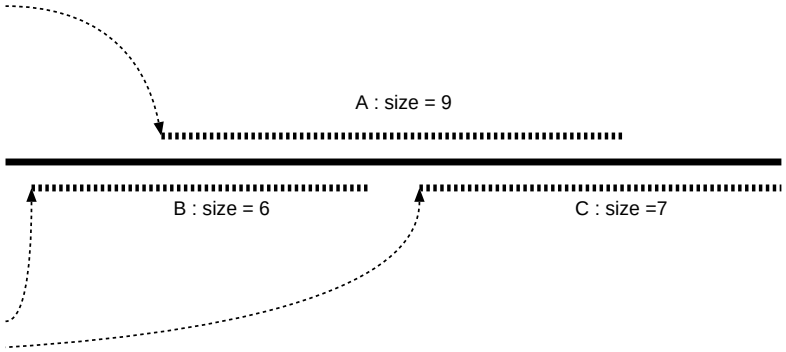
- Break the problem into smaller sub-problems.
- Solve these problems optimally using the 3-step recursively
- Use these optimal solutions to construct an optimal solution for the original problem.

Dynamic Programming (3)

- similar sub-problems
- overlapping sub-problems
- Bottom-Up approach
- Top-Down approach

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.

Dynamic Programming Algorithm

DNAPack

Initialization :

$$BestComp[0] = 0$$

Recurrence :

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

Speed up techniques

- Repeats with common starting substring called *seeds*
- Hash-table on the *seeds*
- $MinCost(j+1, i)$ needs to be computed for j's which $BestComp[j]$ is optimized by a repeat segment.
- 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

- 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 of the first copy operation in a repeat.
- The most frequent base substituted for another base in a repeat substitution.

CODE

- The CODE region consists of two types of segments
  - repeats
  - non-repeats
- There are no two consecutive 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.

The encoding of the repeats(1)

A repeat segment must contain the following information :

- The type of the repeat
- The offset of the origin of the repeat.
- The sequence of operations which transform the reference string into the actual repeat.

The encoding of the repeats(2)

- 0 for the end of the repeat after a Copy
- 1 for a Replace after a Copy
- 0 for a Copy after a Replace
- 1 for a Replace after a Replace

Comparison of Results

sequence	length	BioCompress-2	GenCompress	CTW-LZ	DNACompress	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**
UMDYSTROP	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**
UMHDABCD	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
ANMTPACGA	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**

From Compression to Distance

- Let $Comp(S)$ be the compressed size of string S.
- Let $dist(S,T) = \frac{Comp(ST)+Comp(TS)}{Comp(S)+Comp(T)}$
- Make phylogenetical trees from these distances.

Transformation Distance Problem

- A source string s
- A target string t
- Authorized Operations :
  - $Copy(r)$
  - $ReverseCopy(r)$
  - $Insertion$

Transformation Distance Problem (3)

- The *Transformation Distance* was defined by Varré et al. (1999)

- Their algorithm was based on finding shortest path in a graph represenation of the modeling.

- The complexity is $O(n^4)$ in time and in space.

Transformation Distance (B.B. and J.-M.S 2004)

- Behzadi and Steyaert PSC'2003,SPIRE'2004.

- Direct Dynamic Programming Approach.

- Based on KMP string matching algorithm.

- $O(n^2)$ in time and $O(n)$ in space.

Extended (and Edit) Transformation Distance (B.B. and J.-M.S 2004)

- Point deletions on the copied segments
- $O(n^3)$ in time and $O(n)$ in space.

- Edit Transformation Distance
- $O(n^3)$ in time and $O(n)$

Applications of Comparison of Sequences

- Molecular Biology
- File Comparison
- Text Compression
- Screen Redisplay
- Spell Checking
- Dendrochronology

Last Slide

Nice to meet you

and

Hope to See you Next Year