Efficient data structures for storage and retrieval of multiple biosequences

Stefan Kurtz

Dept. for Genome Informatics
Center for Bioinformatics Hamburg
University of Hamburg

March 27, 2012

Contents

Contents

Sequence representations

- all sequence processing tasks require some form of sequence representation
  - in-memory
  - on-disk (persistent)
- simplest: byte array with one byte per character
- too much for mammalian or plant genomes:
  - human: $\approx$ 3 GB
  - barley: $\approx$ 5 GB
  - wheat: $\approx$ 16 GB
- and too much for NGS-data

Requirements of sequence representations

- space efficiency
  - $\lceil \log_2 \alpha \rceil$ bits/char for sequences over alphabet of size α
- time efficiency
  - constant time sequential and random access to sequence content
- support for multiple sequences
  - chromos. from assembled genomes
  - contig sets from uncompleted genomes
  - short read sets
- alphabet independence
  - not only DNA & proteins
  - IUPAC ambiguity codes
  - user defined alphabets
- support for standard file formats (Fasta, GenBank, EMBL, FASTQ), (un)zipped
- metadata support
  - number of sequences
  - sequence descriptions
  - sequence lengths
  - quality values
  - character distribution
- developer support
  - availability as library
  - scripting language bindings
  - reading directions
    - reverse/forward
    - reverse compl.
  - standard transformations
    - codon translation
    - *k*-mer enumeration

Contents

Encoded sequences

Our solution for representing genomic sequences: *GtEncseq*

in-house use for $\approx$ 5 years, optimized, polished and published this month

A new efficient data structure for storage and retrieval of multiple biosequences

Sascha Steinbiss and Stefan Kurtz

Abstract—Today's genome analysis applications require sequence representations allowing for fast access to their contents while also being memory-efficient enough to facilitate analyses of large-scale data. While a wide variety of sequence representations exist, lack of a generic implementation of efficient sequence storage has led to a plethora of poorly reusable or programming language-specific implementations. We present a novel, space-efficient data structure (GtEncseq) for storing multiple biological sequences of variable alphabet size, with customizable character transformations, wildcard support and an assortment of internal representations optimized for different distributions of wildcards and sequence lengths. For the human genome (3.1 gigabases, including 237 million wildcard characters) our representation requires only $2 + 8 \cdot 10^{-6}$ bits per character. Implemented in C, our portable software implementation provides a variety of methods for random and sequential access to characters and substrings (including different reading directions) using an object-oriented interface. In addition, it includes access to metadata like sequence descriptions or character distributions. The library is extensible to be used from various scripting languages. GtEncseq is much more versatile than previous solutions, adding features that were previously unavailable. Benchmarks show that it is competitive with respect to space and time requirements.

Index Terms—Data storage representations, biology and genetics, software engineering, reusable libraries

1 Introduction

The ever growing size of sequence data from Next Generation Sequencing (NGS) efforts in the form of raw reads, assembled contigs and complete genomes demands for more flexible and efficient software allowing to store this data and facilitating fast access to their contents. For a flexible and efficient sequence representation, we consider a number of desirable features, addressed

absolute sequence positions to relative positions.

Representations of biological sequences of course need to support nucleotide and amino acid alphabets. However, these come in many different variations, like notation for masked positions, uppercase and lowercase notation and with different sets of wildcard symbols (for sequence positions containing ambiguous nucleotides or amino acids). A flexible representation handles these variations in a sensible way. In many cases it is nec-

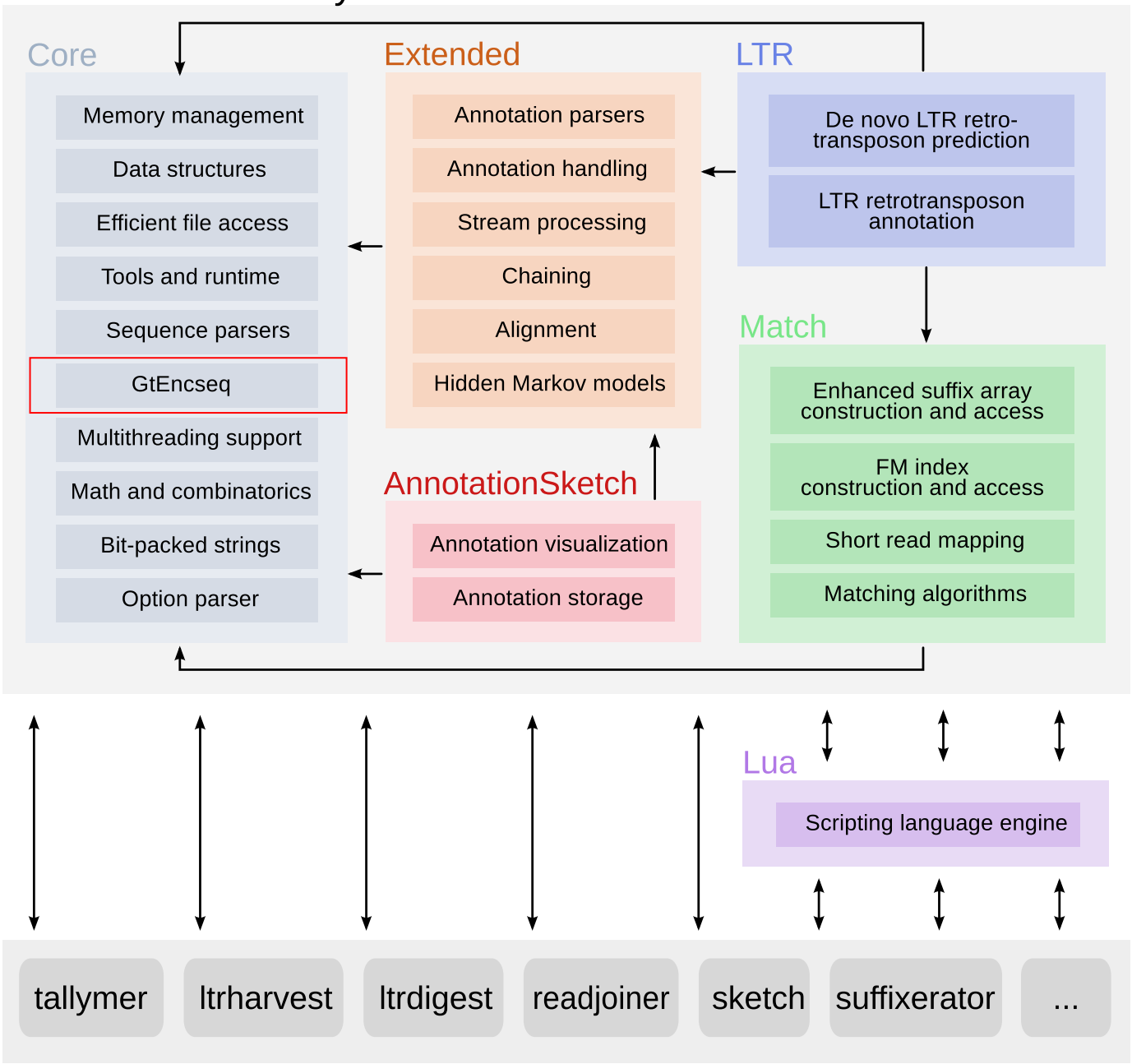
GtEncseq satisfies all mentioned requirements

GtEncseq: available as part of the *GenomeTools* genome analysis software package

GenomeTools (`http://genometools.org`)

- written in portable C for POSIX compliant systems
- UNIX (Linux, BSD, Mac OS X, ...), Windows (with Cygwin)
- open source (BSD-license)
- components:
  1. `libgenometools` shared library
  2. collection of programs ("tools")

GenomeTools library



Tools using the *GtEncseq*

Tallymer: fast and memory-efficient *k*-mer counting

(S. Kurtz *et al. BMC Genomics*, 9:517 (2009))

LTRharvest: de novo detection of LTR retrotransposons

(D. Ellinghaus *et al. BMC Bioinformatics*, 9:18 (2008))

LTRdigest: annotation of internal features of LTR retrotransposons

(S. Steinbiss *et al. Nucleic Acids Research*, 37(21):7002–7013 (2009))

Readjoiner: string graph-based short-read assembly

(G. Gonnella and S. Kurtz. *BMC Bioinformatics*, accepted)

MetaGenomeThreader: gene prediction in metagenome sequences

(D.J. Schmitz-Hübsch and S. Kurtz. In *Metagenomics. Methods in Molecular Biology*, 325–338, Humana Press, Totowa, NJ)

Uniquesub: minimum unique substrings for designing tiling arrays

(S. Gräf *et al. Bioinformatics*, 23(13):i195–i204 (2007))

Previous solutions

SeqAn (Döring et al., BMC Bioinformatics, 2008)

C++, generic programming, compile-time optimizations

BLAT encoding (Kent, Genome Res. 2002)

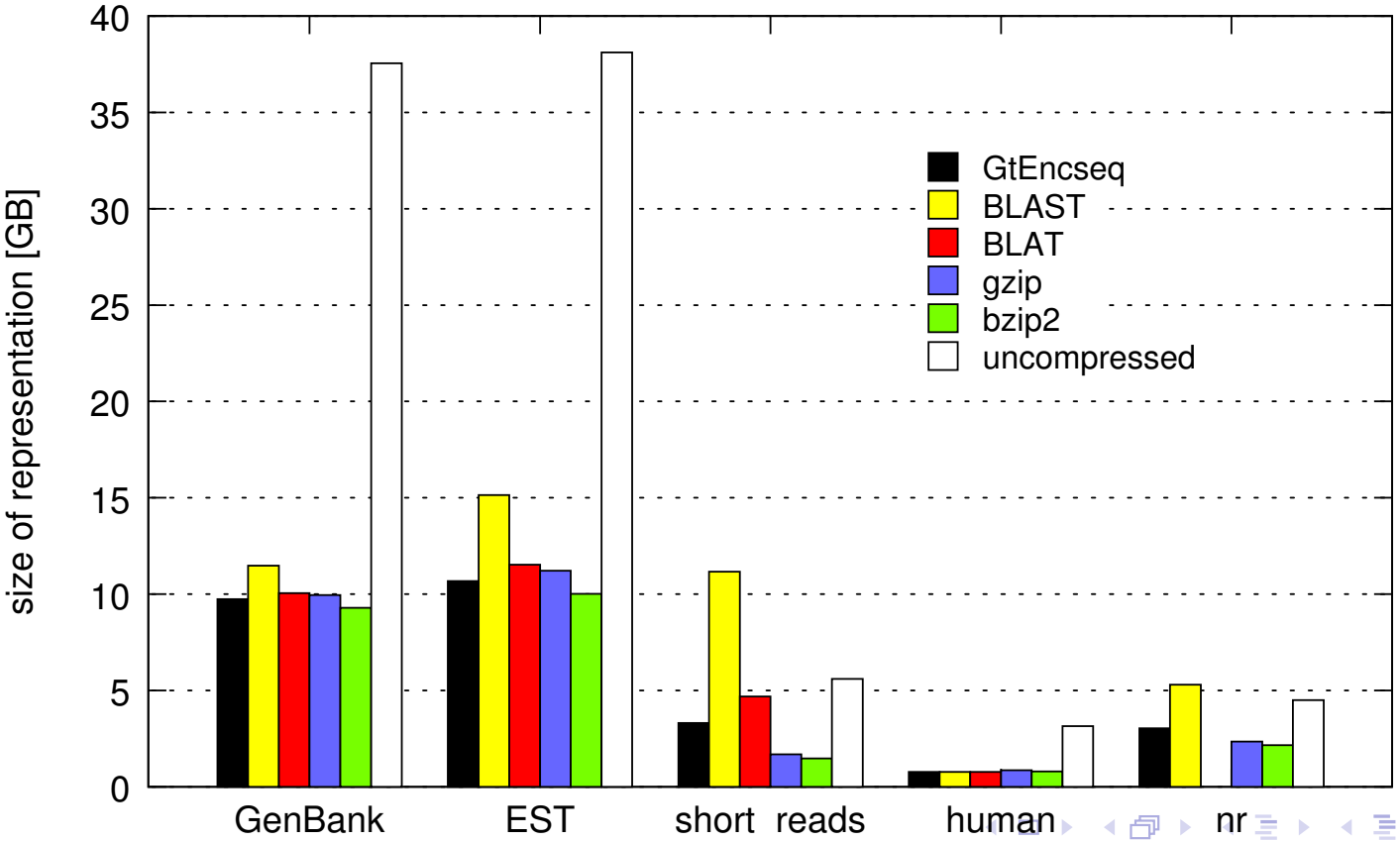
part of BLAT (aka 2bit encoding), only DNA, very simple, no library

BLAST encoding (Altschul et al., 1997)

only DNA and protein sequences, optimized for sequential access,
formatdb/makeblastdb generates the format,
NCBI-toolkit allows to access it

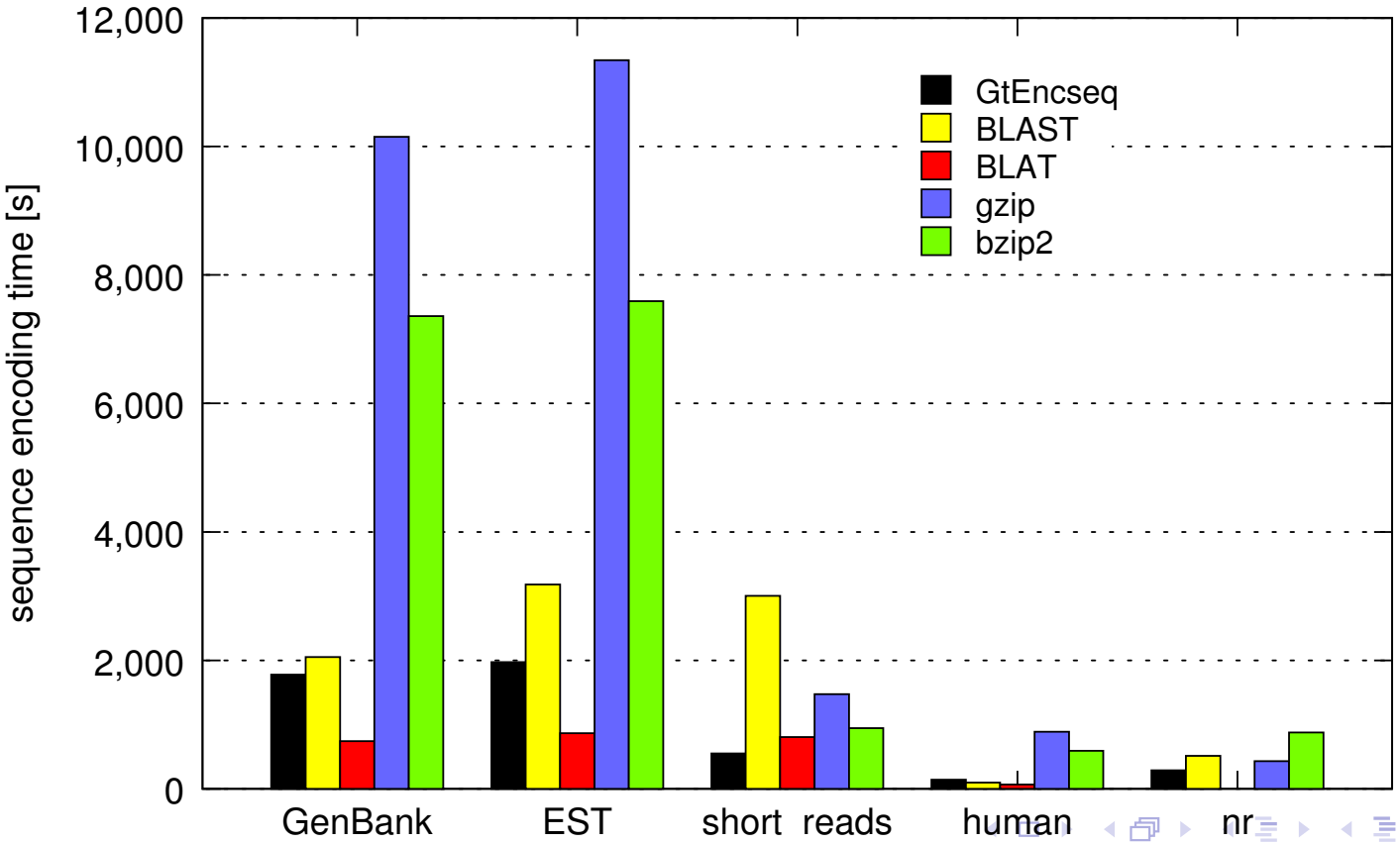
Encoding performance – file size

- GenBank: 37.54 GB DNA sequences
- EST: 38.11 GB DNA sequences
- short reads: 5.6 GB DNA reads, no wildcards, 35 bp
- human: DNA, human genome build 37, 3.14 GB
- nr: 4.49 GB protein sequences



Encoding performance – encoding time

- GenBank: 37.54 GB DNA sequences
- EST: 38.11 GB DNA sequences
- short reads: 5.6 GB DNA reads, no wildcards, 35 bp
- human: DNA, human genome build 37, 3.14 GB
- nr: 4.49 GB protein sequences



GtEncseq access performance

Benchmarking scenario:

(1) Extraction of all exonic sequences from the human genome

(2) 10^6 random accesses to single bases

	GtEncseq	SeqAn		BLAT enc.	BLAST enc.
Version	1.3.8	1.2		v34	6.1
Implementation language	C	C++		C	C
		packed	external		
Sequence loading (s)	0.003	128.1	207.3	0.001	0.085
Extraction					
(1) Exonic sequences (s)	6.5	5.2	307.3	5.3	362,644
(2) Random access (s)	0.3	0.2	0.4	719.0	segfault
Memory peak (MB)	737	951	365	3,184	1,458

Poor behavior is shown in red

Conclusion for this part:

- Space and time requirements: *GtEncseq* is competitive with the best tool in each category
- *GtEncseq* is by far the most versatile and complete solution

Contents

Data types

Sequencing data

- newly sequenced reads obtained using NGS technology
- often given in FASTQ format (P. Cock *et al. Nucleic Acids Res*, 38(6):1767–71 (2009))
  - descriptions
  - sequences
  - quality values (e.g. encoded error probabilities)

Resequencing data

- short reads mapped to an established reference sequence
  $\Rightarrow$ essentially set of alignments
- often given in BAM/SAM format (H. Li *et al. Bioinformatics*, 25:2078–9 (2009))
  - reference position
  - read number
  - alignment edit operations

Intention

Disclaimer

The methods of this part are not completely new

Intention

Our work aims at providing...

1. ...an integrated solution for sequence storage and access
   - without qualities ($\Rightarrow$ *GtEncseq*)
   - with qualities (ongoing work)
2. ...a unified interface
3. ...a set of reusable code modules for integrating previously isolated methods

Sequencing data – FASTQ input format

```
@read.1 length=26
GAAACATTACCAGTTCTGTTTCATTT
+
IIIIIIIIIIII<I&DIIIIIEF--I
@read.2 length=26
TACAGATGACCAGTTAAGGGCAATCT
+
8BBABABBABB8:!B:DBB!!!###!
```

Selection of specific encoding techniques

- description lines are highly repetitive
  - increasing numbers, equal strings, ...
  - analyze structure and apply most appropriate encoding scheme
    (S. Deorowicz *et al. Bioinformatics*, 27(6):860–2 (2011))
- sequence/quality pairs have characteristic occurrence frequencies
  - employ encoding scheme based on statistical measures
    (W. Tembe *et al. Bioinformatics*, 26(6):2192–94 (2010))

Sequencing data – Decompression

- sequential access to whole read set
  - decode read set file from the start
  - fast, but inefficient for retrieval of single arbitrary reads
- random read access requires sampling
  - for every dth read store the starting position of its encoding
  - d is *sampling distance* providing time/space tradeoff
  - small sampling distance $\Rightarrow$ fast random access, but large space requirement

Sequencing data – Results

SRR029844.1.filt, 1000 genomes proj, 5.7×10^7 reads, 76 bp, 1.14 GB					
sampling distance	bits/pair	bits/desc	compr. ratio	compression speed	extraction time for 10^5 reads
∞	5.53	0.55	3.81	10.37 MB/s	78394 s
100000	5.53	0.56	3.80	10.55 MB/s	2080 s
10000	5.57	0.61	3.75	10.46 MB/s	221 s
1000	5.96	1.10	3.33	10.55 MB/s	24 s
100	9.85	6.04	1.56	10.27 MB/s	7 s

Storage of resequencing data

Basic approach

- store differences between reads and reference
- compress data by encoding edit operations

(M.H.-Y. Fritz *et al.*, *Genome Res* 21(5):734–40 (2011))

Input

- reference sequence
- sorted alignments of reads to reference ("mapped reads")
  - SAM (Sequence Alignment/Map, tab-delimited text file) or
  - BAM (binary version of SAM)

Output

- compressed representation allowing to extract all mapped reads (including quality values) given the reference sequence

Resequencing data – Example

```
ACGATCTTAATGCCTTACTTGTT--GG-CATTC      reference

  ATCGTAAT---TTAC
        ATGCCTTACTTGT                  mapped reads
              ACTTGTTATGGCC
```

position	mismatches	insertions	deletions
3	3: T → G	-	4: 3
6	-	-	-
7	-	7: AT, 3: C	-

Resequencing data – Encoding results

Options for quality storage

- none/of variations only/all qualities

Encoding results

- 6.4 GB Illumina reads from 1000 Genomes project mapped to human chromosome 20 reference
- reference stored as *GtEncseq*

preserved information	bits/base	compression speed	decompression speed
sequence, strand	0.63	13.58 MB/s	32.60 MB/s
sequence, strand, qualities of variations	1.16	13.25 MB/s	29.01 MB/s
sequence, strand, all qualities	5.21	11.12 MB/s	15.42 MB/s

Future Work

GtEncseq – storage of sequence variants

- nonredundantly store a set of similar genomic sequences
  - e.g. sets of individual genomes, strains, ...
  - access via virtual concatenation

GtEncseq – scripting language bindings

- improve efficiency of scripting language bindings
- introduce proper Perl bindings (*ctypes*)

GenomeTools integration

- unified object-oriented interface and command-line tools available for *GtEncseq* and short read processing
  - creation, loading, access

Acknowledgements

- Sascha Steinbiss (main developer, interfaces, integration)
- Dirk Willrodt (header compression, integration of code)
- Joachim Bonnet (short read compression)
- Giorgio Gonnella (large-scale testing)