GTRAC: Supplementary Data

April 5, 2016

In this supplementary data we provide some more implementation details of the GTRAC (GenoType Random Access Compressor) algorithm. We first provide instructions on using the GTRAC source code, and then discuss some implementation details. Finally, we provide some additional results.

Getting Started

Install

To install GTRAC, follow the following instructions:

```
# download GTRAC
git clone https://github.com/kedartatwawadi/GTRAC

# Run the install file to download and install
# rsdic,TGC and GTRAC libraries
./install.sh
```

The **install.sh** script downloads and installs the following libraries, before compiling the GTRAC code.

1. **rsdic** (`https://github.com/kedartatwawadi/rsdic`):
   The library implementation of succinct bitvectors.
2. **TGC** (`https://github.com/refresh-bio/TGC`):
   The TGC Compressor, which we use for VCF file handling and variant dictionary compression.

Run GTRAC

One can run an example of the GTRAC compressor by running the following code:

```
# Run GTRAC for the chromosomes mentioned in config.ini
# of the 1000GP project
./run -abceik
```

The **run** shell script does the following:

1. Downloads VCF and reference FASTA files (for chromosome 22, by default) corresponding to the 1000 Genome Project.

2. Uses tools from the **TGC** repo to process the VCF files and convert them into the $(\mathcal{H}, \mathcal{V}_\mathcal{D})$ representation.

3. Uses GTRAC to compress the binary matrix $\mathcal{H}$.

4. Uses tools from the **TGC** repo to compress the variant dictionary $\mathcal{V}_\mathcal{D}$.

Test Random Access

GTRAC supports per-variant extraction (column-wise extraction), and per-haplotype extraction (row-wise extraction) from the compressed symbol matrix $\mathcal{H}_K$. These features can be tested by running the following examples:

```
# Move to the correct directory
cd ../Data/chr22

# single-variant extraction:
# This corresponds to extracting 792-799th variant (the 100th symbol)
# information at a time.
../../GTRAC/src/gtrac_decomp c chr22.list 100

# complete haplotype extraction:
# This corresponds to extracting 1000th haplotype
# from the compressed dataset.
../../GTRAC/src/gtrac_decomp f chr22.list 1000

# haplotype sub-sequence extraction:
# This corresponds to extracting a subsequence of length 1000
# starting from 10th symbol of 800th haplotype of the compressed dataset.
../../GTRAC/src/gtrac_decomp f chr22.list 800 10 1000
```

Datasets

We used two large datasets; the 1000 Genome Project Phase 1 *H.Sapiens* dataset and the 1001 Genomes *A.Thaliana* dataset. These datasets can be accessed at the follwing locations:

H. Sapiens Dataset

The 1000 Genome Project phase 1 can be downloaded from:

1. Link to the Reference files for all the chromosomes:
`ftp://ftp.ncbi.nlm.nih.gov/genomes/archive/old_genbank/Eukaryotes/vertebrates_mammals/Homo_sapiens/GRCh37/Primary_Assembly/assembled_chromosomes/FASTA/`.

2. Link to the VCF files for all the chromosomes:
`ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase1/analysis_results/integrated_call_sets/`

A.Thaliana dataset

The *A.Thaliana* dataset comes from 4 different subprojects, containing in total 775 sequenced strains.

1. Link to the Reference FASTA files for all chromosomes :
   `ftp://ftp.arabidopsis.org//Sequences/whole_chromosomes/`
2. The VCF files can be downloaded from the 4 different subprojects:
   (a) **MPICao2010—80 Arabidopsis Thaliana accessions**:
   `http://1001genomes.org/data/MPI/MPICao2010/releases/2012_03_13/strains/`
   (b) **Salk—Arabidopsis Thaliana strains sequenced by the Salk Institute**:
   `http://1001genomes.org/data/Salk/releases/2011_06_28/TAIR10/strains/`
   (c) **GMINordborg2010—Arabidopsis Thaliana strains sequenced by the Gregor Mendel Institute**:
   `http://1001genomes.org/data/GMI/GMINordborg2010/releases/2011_08_04/strains/`
   (d) **MPICWang2013—343 Arabidopsis Thaliana accessions**:
   `http://1001genomes.org/data/MPI/MPICWang2013/releases/2013_04_15/strains/`

As the datasets are the same as the ones used by **TGC**, more information about their access can be obtained from `sun.aei.polsl.pl/tgc/`.

Additional Compression Details

The variant dictionary $\mathcal{V}_\mathcal{D}$ represents an indexed list of all the variants present in the dataset. Note that the variant dictionary typically consists of a small portion of the memory usage of the $(\mathcal{H}, \mathcal{V}_\mathcal{D})$ representation (e.g., 5% to 10% for the 1000GP and the *A.Thaliana* datasets). Thus we concentrate concentrate our efforts on the compression of the binary matrix $\mathcal{H}$, and use the same technique as the one proposed in **TGC** (`sun.aei.polsl.pl/tgc/`) to compress the variant dictionary $\mathcal{V}_\mathcal{D}$.

The basic idea on the compression of the variant dictionary $\mathcal{V}_\mathcal{D}$ is to compress each type of variant separately. The different types of variants are Single Nucleotide Polymorphism (SNP), insertions, deletions, and Structural Variants (SV). For each variant type, the variants' positions are differentially encoded (e.g., distances between consecutive SNPs). The remaining parameters of each of the variants are also encoded. For example, for SNPs, the substituting symbol is stored, for insertions, the length and the inserted symbols are stored, etc. All these values are encoded using a variant of arithmetic coding with appropriate contextual models. More details can be found in the the TGC paper.

The method to compress the binary matrix is explained in the main paper. The implementation uses hash tables to find the matches and construct the phrases. The time required to compress the binary matrix $\mathcal{H}$ for each chromosome is on average about 30 minutes for the *H. Sapiens* dataset, and about 15 minutes for the *A. Thaliana* dataset.

Additional Results & Discussions

We experimented with different values of the parameter K for forming the symbol matrix $\mathcal{H}_{\mathcal{K}}$. For K being multiples of 8, it was observed that memory handling is much easier. The random access is also effectively faster due to the byte-aligned memory handling. We experimented with values $K = 8$ and $K = 16$. For $K = 16$, the overall archival memory usage is similar to $K = 8$ (slightly higher). $K = 16$ results in faster compression and haplotype-extraction (with an increase of almost a factor of 2). However, the single-variant extraction times remain almost the same. This is expected as our haplotype substring extraction algorithm extracts the sub-string a symbol at a time. Thus, increase in symbol size results in lesser symbols to be extracted for a sub-string.

The comparison for symbol-size 8 and 16 are summarized in Table 1. The e parameter encoding requires less memory mainly due to reduced row-size of the symbol matrix. The C parameter encoding takes more amount of memory, as each symbol we store is now 2-bytes instead of 1-byte. For higher values of K, the total amount of memory required increases significantly, mainly due to increase in the symbol size (which in turn increase C parameter encoding memory requirement).

K parameter comparison for chromosome 22 of the *H.Sapiens* dataset

	GTRAC, $K = 8$	GTRAC, $K = 16$
Total Archive Memory	16MB	17MB
C parameter encoding	4.2MB	7MB
e parameter encoding	6.9MB	4.9MB
Per genome decompression Time	0.17s	0.07s
Per Variant decompression Time	15ms	17ms

Table 1: Comparison of performance for $K = 8$ and $K = 16$