GTC
Supplementary material

Agnieszka Danek and Sebastian Deorowicz

December 15, 2017

Contents

1 Method

Detailed description of construction of GTC archive is presented in Fig. 1. It is an extended version of a similar figure from the main paper.

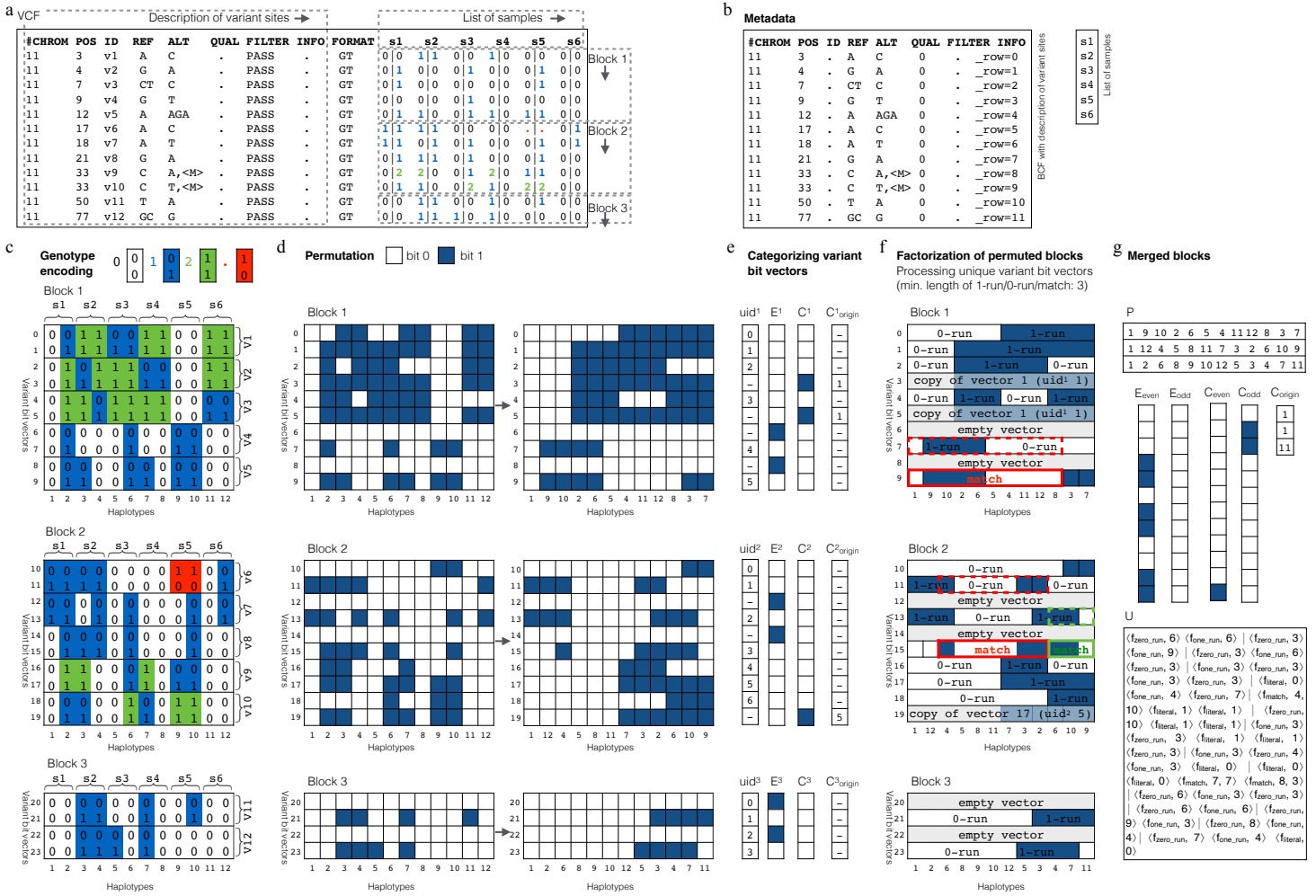

Figure 1: Construction of GTC archive. (**a**) The input VCF file with genotypes of 6 diploid samples at 12 variant sites. The VCF file is decomposed into metadata and blocks of genotypes. Each block (here: max. 5 variant sites) is then processed separately. (**b**) Metadata: variant sites description (site-only BCF) and list of samples. (**c**) Bit vector representation of genotypes. Each variant site is described by two bit vectors (e. g. v1 by variant bit vectors 0 and 1). The genotype of a haplotype, at each variant site, is described by a dibit (two bits located in two successive variant bit vectors). (**d**) Permutation of haplotypes in all blocks. (**e**) Categorizing variant bit vectors. Set bit in vector E^i or C^i indicates that corresponding variant bit vector in block i is empty or copied, respectively (here E and C are not divided into *odd* and *even* parts). Ids of unique vectors are shown in uid^i array. The C^i_{origin} stores *uid*s of original vectors for successive copied vectors. (**f**) Factorization of all permuted blocks. Empty and copied vectors are marked. All unique bit vectors are described as a sequence of tuples: longest possible 0-runs, 1-runs, matches to previous bit vectors in the block and literals. (**g**) All blocks are merged in their order in the input VCF file. The permutations P byte array and the bit vectors $E_{odd/even}$ and $C_{even/even}$ are formed. The size of C_{origin} vector is equal to the number of copied vectors, while the *uid*s it stores are adjusted while merging successive C^i_{origin} vectors. The U byte array stores all variant bit vectors (within all blocks) encoded into tuples (adjusting match positions as in C_{origin}). Vertical lines indicate the start of successive variant bit vectors (these positions are stored in U_{pos} array).

2 Examined programs

2.1 General parameters of programs

The following programs were used in the experimental part.

- BGT v. 1.0-r283-dirty
- GQT v. 1.1.3
- SeqArray v. 1.14.1 (with gdsfmt v. 1.10.1)
- GTRAC v. 0.1.0-fix
- GTC v. 1.1

2.2 Exact command line parameters

The command lines given below were used to obtain the results depicted in figures in the main text as well as in the supplementary material. The `archive` means the name of the compressed file. The `input.vcf.gz` means the name of the input gzipped VCF file used as an input. To measure the real time of supporting the queries (not affected by the disk speed) the times were obtained when the output was sent to `/dev/null` as marked below.

In all cases, except for SeqArray, the time was measured using `/usr/bin/time -v` command. For the SeqArray case, the time was measured inside the R code (using `system.time` function) to exclude the time of loading the SeqArray library.

BCFtools

- Decompression:

```
./bcftools view -Ou > /dev/null
```

- Single variant query:

```
./bcftools view -Ou -r <chr>:<start_pos>-<start_pos> input.vcf.gz > /dev/null
```

- Range of variants query:

```
./bcftools view -Ou -r <chr>:<start_pos>-<end_pos> input.vcf.gz > /dev/null
```

- Sample query:

```
./bcftools view -Ou -s <sample_name> > /dev/null
```

- Many samples query:

```
./bcftools view -Ou -S <sample_file_name> > /dev/null
```

- Many samples query for given range of variants:

```
./bcftools view -Ou -S <sample_file_name> -r <chr>:<start_pos>-<end_pos> > /dev/null
```

BGT

- Compression:

```
./bgt import -S -o archive input.vcf.gz
```

- Decompression:

```
./bgt view -b -u archive > /dev/null
```

- Single variant query:

```
./bgt view -b -u -r <chr>:<start_pos>-<start_pos> archive > /dev/null
```

- Range of variants query:

```
./bgt view -b -u -r <chr>:<start_pos>-<end_pos> archive > /dev/null
```

- Sample query:

```
./bgt view -b -u -s,<sample_name> archive > /dev/null
```

- Many samples query:

```
./bgt view -b -u -s <sample_file_name> archive > /dev/null
```

- Many samples query for given range of variants:

```
./bgt view -b -u -s <sample_file_name> -r <chr>:<start_pos>-<end_pos> archive >
    /dev/null
```

GQT

- Compression:

```
./gqt convert bcf -i input.vcf.gz
```

SeqArray

- Compression:

```
library(SeqArray)
seqVCF2GDS("input.vcf.gz", "archive.gds", storage.option="LZMA_RA")
```

- Decompression:

```
library(SeqArray)
f <- seqOpen("archive.gds")
seqGDS2VCF(f, "/dev/null")
seqClose(f)
```

- Single variant query:

```
library(SeqArray)
(gds.fn <- "archive.gds")
(f <- seqOpen(gds.fn))
seqSetFilterChrom(f, <chr>, from.bp=as.numeric(<start_pos>), to.bp=as.numeric(<start_pos>))
seqGDS2VCF(f, "/dev/null")
seqClose(f)
```

- Range of variants query:

```
library(SeqArray)
(gds.fn <- "archive.gds")
(f <- seqOpen(gds.fn))
seqSetFilterChrom(f, <chr>, from.bp=as.numeric(<start_pos>), to.bp=as.numeric(<end_pos>))
seqGDS2VCF(f, "/dev/null")
seqClose(f)
```

- Sample query:

```
library(SeqArray)
(gds.fn <- "archive.gds")
(f <- seqOpen(gds.fn))
(samp.id <- seqGetData(f, "sample.id"))
seqSetFilter(f, sample.sel=which(samp.id==<sample_name>))
seqGDS2VCF(f, "/dev/null")
seqClose(f)
```

- Many samples query:

```
library(SeqArray)
(gds.fn <- "archive.gds")
(f <- seqOpen(gds.fn))
(samp.id <- seqGetData(f, "sample.id"))
s_ids = read.table(sample_file_name)
ww = c()
for(i in 1:no_samples)
    ww = c(ww, which(samp.id==s_ids[i,]))
seqSetFilter(f, sample.sel=ww)
seqGDS2VCF(f, "/dev/null")
seqClose(f)
```

- Many samples query for given range of variants:

```
library(SeqArray)
(gds.fn <- "archive.gds")
(f <- seqOpen(gds.fn))
(samp.id <- seqGetData(f, "sample.id"))
s_ids = read.table(sample_file_name)
ww = c()
for(i in 1:no_samples)
    ww = c(ww, which(samp.id==s_ids[i,]))
seqSetFilter(f, sample.sel=ww)
seqSetFilterChrom(f, <chr>, from.bp=as.numeric(<start_pos>), to.bp=as.numeric(<end_pos>))
seqGDS2VCF(f, "/dev/null")
seqClose(f)
```

GTRAC

- Compression:

```
./gtrac_comp <file_file_names> archive
```

- Single variant query:

```
./gtrac_decomp c archive <variant_id> /dev/null
```

- Sample query:

```
./gtrac_decomp f archive <sample_id> /dev/null
```

GTC

- Compression:

```
./gtc compress -t 1 -o archive input.vcf.gz
```

- Decompression:

```
./gtc view -b -c 0 archive > /dev/null
```

- Single variant query:

```
./gtc view -b -c 0 -r <chr>:<start_pos>-<start_pos> archive > /dev/null
```

- Range of variants query:

```
./gtc view -b -c 0 -r <chr>:<start_pos>-<end_pos> archive > /dev/null
```

- Sample query:

```
./gtc view -b -c 0 -s <sample_name> > /dev/null
```

- Many samples query:

```
./gtc view -b -c 0 -s @<sample_file_name> > /dev/null
```

- Many samples query for given range of variants:

```
./gtc view -b -c 0 -s @<sample_file_name> -r <chr>:<start_pos>-<end_pos> >
    /dev/null
```

GTC for GTRAC comparison

- Compression:

```
./gtc compress_dev p -o archive input.vcf.gz
./gtc compress_dev c archive
```

- Single variant query:

```
./gtc view_dev -j <variant_id> archive > /dev/null
```

- Sample query:

```
./gtc view_dev -i <sample_id> archive > /dev/null
```

3 Environment

The workstation used for the experiments:

- 2 Intel Xeon E5-2670 CPUs; 12 cores per CPU, each clocked at 2.3 GHz,
- 128 GB RAM,
- 2 Seagate Enterprise NAS HDD of size 6 TB each in RAID0 configuration; `hdparm -t` reported speed: 300 MB/s.

4 Additional experiments and results

4.1 Data sets

The 1000 Genomes Project — Phase 1

The 1000 Genome Project data sets were downloaded from:
`ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase1/analysis_results/integrated_call_sets/`

The 1000 Genomes Project — Phase 3

The 1000 Genome Project data sets were downloaded from:
`ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/`

The Haplotype Resource Consortium

The HRC data sets were downloaded from the European Genom-phenom Archive (EGAS00000000029).

Sampled HRC

The sampled HRC data sets were obtained from the HRC Chromosome 11 data by randomly picking 1000, 2000, 3000, 4000, 5000, 7000, 10000, 15000, and 20000 samples.

4.2 Adjusting GTC parameters

Various parameters of GTC were adjusted on the 1000 Genomes Project Phase 3 Chromosome 11 data (2504 samples). The results are presented in Figures 2–7.

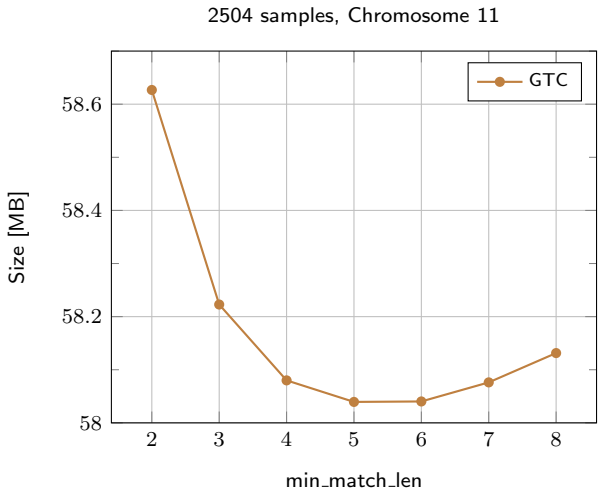

Figure 2: Influence of the minimal match length

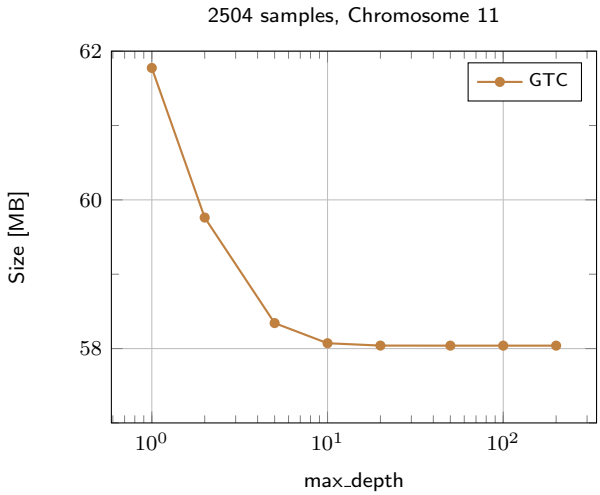

Figure 3: Influence of the maximal allowed depth of matches

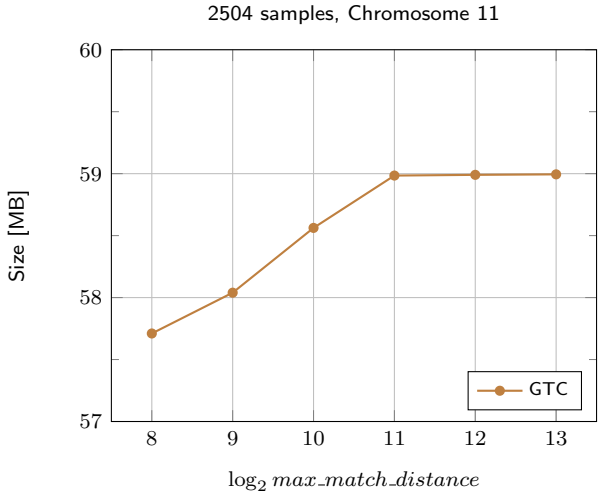

Figure 4: Influence of the maximal match distance

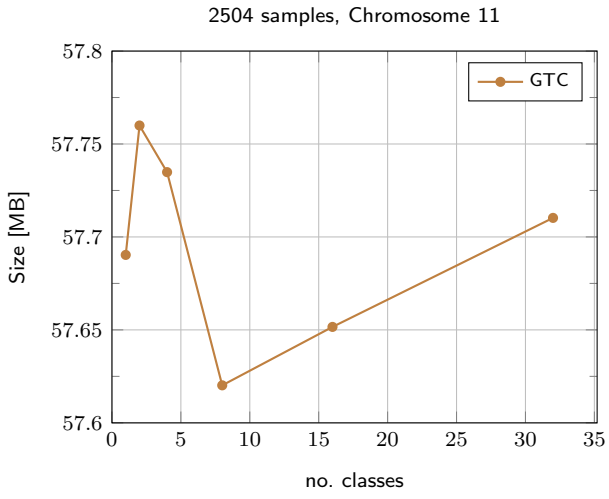

Figure 5: Influence of the number of classes

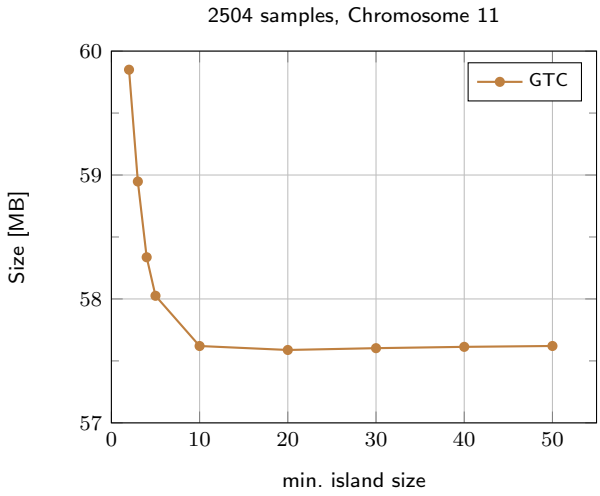

Figure 6: Influence of the minimal island of literals size

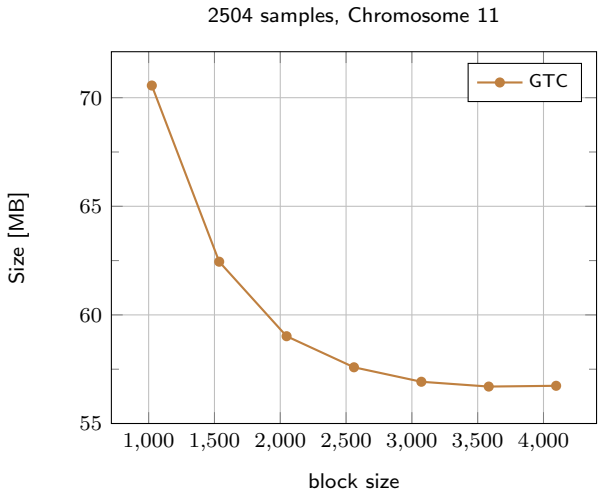

Figure 7: Influence of the block size

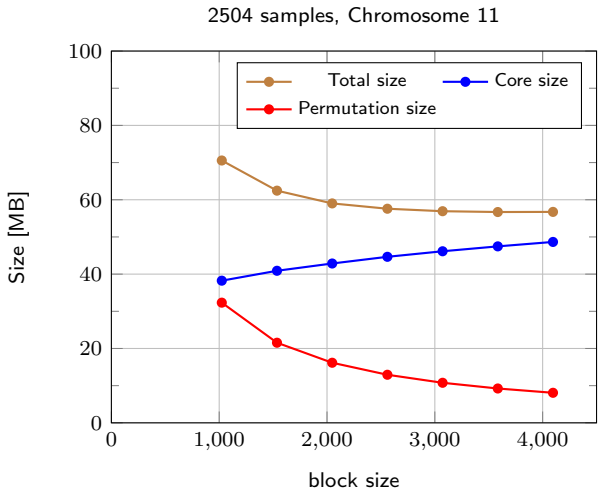

Figure 8: Sizes of components: permutation description and core (description of matches, 0-runs, etc.) for various block sizes

According to the results the following parameters were used in the final implementation:

- minimal match length — 5,
- maximal allowed depth of matches — 100,
- maximal distance (offset) for matches — 256,
- number of classes — 8,
- minimal island of literals size — 20,
- block size — 3584.

4.3 Sizes of compressed archives

Table 1 shows numerical results presented in Fig. 2a in the main manuscript.

Table 1: Sizes of compressed sampled archives of HRC Chromosome 11 data.

Collection size	Size [MB]				
(no. of sampl.)	BGT	GTC	VCF.gz	GQT	SeqArray
1,000	55	25	191	176	35
2,000	68	34	334	319	50
3,000	83	42	472	464	66
4,000	94	49	604	611	81
5,000	105	57	735	759	96
7,000	127	69	990	1,055	126
10,000	158	88	1,385	1,504	172
15,000	209	117	1,989	2,261	251
20,000	259	146	2,581	3,022	333
27,165	331	185	3,425	4,115	617