GREDU User Reference Manual

Mingfu Shao *

Laboratory for Computational Biology and Bioinfomratics,
École Polytechnique Fédérale de Lausanne (EPFL),
Lausanne, Switzerland

October 7, 2018

1 Introduction

GREDU (Genome REarrangements with DUplications) is a software package implemented several exact algorithms to compute the edit distances under various evolutionaly models for two genomes with duplicate genes. Specifically, GREDU contains the following five programs:

1. **dcj**, implements an exact algorithm to compute the DCJ (Double-Cut-and-Join) distance between two genomes with duplicate genes [1].
2. **segdcj**, implements an exact algorithm to compute the edit distance under segmental duplications and DCJ operations for two genomes with duplicate genes [2].
3. **exemplar**, implements an exact algorithm to compute the exemplar breakpoint distance for two genomes with duplicate genes [3].
4. **intermediate**, implements an exact algorithm to compute the intermediate breakpoint distance for two genomes with duplicate genes [4].
5. **maxmatching**, implements an exact algorithm to compute the maximum-matching breakpoint distance for two genomes with duplicate genes [4].

2 Installation

To install GREDU, you need to download two libraries (BOOST and GUROBI), setup the corresponding environmental variables, and then compile the source code of GREDU.

*shaomingfu@gmail.com

2.1 Install BOOST

Download BOOST from `http://www.boost.org`. Uncompress it somewhere (compiling and installing are not necessary). Set environment variable `BOOST_HOME` to indicate the directory of BOOST. For example, for Unix platforms, add the following statement to the file `~/.bash_profile`:

```
export BOOST_HOME="/directory/to/your/boost/boost_1_60_0"
```

2.2 Install GUROBI

Download GUROBI from `http://www.gurobi.com/` and uncompress the package somewhere (compiling and installing are not required). You need to apply an academic license to use the full features of GUROBI (Please refer to the GUROBI documentation for more information.) After that, set two environment variables, `GUROBI_HOME` and `GRB_LICENSE_FILE`, which indicates the directory of GUROBI, and the location of your license file, respectively. For example, for Unix platforms, add the following two statements to the file `~/.bash_profile`:

```
export GUROBI_HOME="/directory/to/your/gurobi/linux64"
export GRB_LICENSE_FILE="/your/license/gurobi.lic"
```

2.3 Compile GREDU

Get the source code of GREDU through `git`:

```
$git clone git@github.com:shaomingfu/grudo.git .
```

Compile the libraries, main source code, and tools through:

```
$lib/build.sh
$src/build.sh
$tools/build.sh
```

After that all executable files (`dcj`, `segdcj`, `exemplar`, `intermediate` and `maxmatching`) will be present at `bin`.

3 Command line

All five programs use the same parameters. Take `exemplar` as an example:

```
$./exemplar <genome1> <genome2> <ILP-time-limit>
```

The first two arguments spcifies two files in which the two genomes are encoded. The third argument specifies the time limit (in seconds) for the GUROBI solver. A set of input-file examples are provided under `bin`. For program `dcj`, you can use `human.dcj` and `mouse.dcj` as input files. For other four programs, you can use `human.all` and `mouse.all` as input files.

4 Input Format

The structure of the genome file is as follows. A genome contains several linear or circular chromosomes, and each chromosome consists of a sequence of genes in the order of their location on the chromosome. Each chromosome contain several lines, and each line specifies a gene, containing four fields separated by spaces.

1. The first field is a *string*, which species the *identifier* of this gene. The identifier should be unique for each gene.
2. The second field is an *signed integer*, which species the *family* of this gene. Genes in the same gene family should have the same absolute value. The orientation of this gene is specified by the sign of this integer.
3. The third field is a *string*, which species the *chromosome name* of this gene.
4. The fourth field is a *integer* choosing from $\{1, 2\}$, where 1 means this chromosome is linear and 2 means this chromosome is circular.

NOTE: Make sure that for program `dcj`, for each gene family, the number of genes in each genome in this gene family are equal (there is no such requirement for the other four programs).

5 Output Format

A file with name `mapping` will be generated in the current directionary specifying the optimal one-to-one correspondence of the genes in the two genomes, in which the identifiers of the genes are used. For program `segdcj`, an additional file `segments` will be generated illstruting the optimal segmental duplications for each genome. The optimal edit distance between the two given genomes will be displayed at the bottom line of the standard output.

6 Tools

If you use the data from Ensembl `http://www.ensembl.org`, we have provided some perl scripts at `tools/ensembltool` to generate the input files for these programs from raw downloaded data.

To download gene order data from Ensembl, use the `customise your download` page. Take human genome as an example. First, choose database (`Ensembl Genes 74`). Then choose dataset (`Homo sapiens genes`). Second, in the Filters, choose proper chromosomes in the `REGION` field (1-22, X and Y), and choose `protein coding genes` in the `GENE` field if necessary. Third, in the Attributes,

choose `Ensembl Gene ID`, `Ensembl Transcript ID`, `Chromosome Name`, `Strand`, `Transcript Start`, `Transcript End` in the `GENE` field, and `Ensembl Protein Family ID(s)` in the `PROTEIN DOMAINS` field. Make sure that these attributes are selected in the same order described above. Last, come to the `Results` tab and save them to a CSV file.

After downloading the data, for example, human genome and mouse genome, we can use the perl script `tools/ensembltool/build.pl` to transform to the required format:

```
$./build.pl <human.list> <mouse.list> <human.input> <moust.input>
```

The first two parameters are the names of the two raw data files, and the last two parameters are the names of the input files of GREDU with the required format. `build.pl` calls the other three scripts in the same directory, where `longest.pl` is to select the longest transcript for each gene, `family.pl` is to select those gene families with the same number of genes in each genome, and `join.pl` is to transform those genes in the selected families to the required format.

NOTE: for program `dcj`, you have to uncomment a few lines of `family.pl` to generate the correct input files (please follow the instruction on line 47). This is because `dcj` requires that for each gene family, exactly the same number of genes should be given. The modified `family.pl` will only keep those gene families with the same gene copy numbers and remove others.

References

[1] M. Shao, Y. Lin, and B.M.E. Moret. An exact algorithm to compute the DCJ distance for genomes with duplicate genes. In *Proc. 18th Int'l Conf. Comput. Mol. Biol. (RECOMB'14)*, volume 8394 of *Lecture Notes in Comp. Sci.*, pages 280–292, 2014.

[2] M. Shao and B.M.E. Moret. Comparing genomes with rearrangements and segmental duplications. *Bioinformatics*, 31(12):i329–i338, 2015.

[3] M. Shao and B.M.E. Moret. A fast and exact algorithm for the exemplar breakpoint distance. In *Proc. 19th Int'l Conf. Comput. Mol. Biol. (RECOMB'15)*, volume 9029 of *Lecture Notes in Comp. Sci.*, pages 309–322, 2015.

[4] M. Shao and B.M.E. Moret. On computing breakpoint distances for genomes with duplicate genes. In *Proc. 20th Int'l Conf. Comput. Mol. Biol. (RECOMB'16)*, volume 9649 of *Lecture Notes in Comp. Sci.*, pages 189–203, 2016.