Documentation of the Chromosome Files
(MEM and AKS Files)
GLEAM

Version V1.3

Wilfried Jakob
KIT, Campus North, Institute for Automation and Applied Informatics (IAI)
Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany
email: wilfried.jakob@partner.kit.edu

Contents

1 Introduction....3
2 Structure of a Chromosome Text File up to (Hy)GLEAM Version 2.2.1....3
2.1 Structure of the elements....4
2.1.1 Header Element....4
2.1.2 Header Parameter Element....4
2.1.3 Segment Descriptors....4
2.1.4 Gene Elements....4
2.2 Examples....5
3 Structure of a Chromosome Text File from (Hy)GLEAM Version 2.2.2....5
3.1 Gene Elements....6
3.2 Examples....6
4 Literature....7

Release Notes

Changes to V1.0:

Introduction

Changes to V1.1:

1. Introduction
2. As of GLEAM version 2.2.2 (from 28.6.2019), only those gene parameters specified in the gene model are written and read for textual chromosome files. This means that older textual chromosome files from version 2.2.2 on can no longer be read! This does not affect binary chromosome files.
3. Behavior with set compiler switch `WITH_DYN_GENES`.

Changes to V1.2:

Editorial adjustments to the format of the file documentations

1 Introduction

Chromosome files are available as binary and text files, the latter being originally intended for export across computer and/or BS boundaries. Both file formats can be loaded interactively, whereby the file format must be specified during the loading process. When initially loading a chromosome file by specification in the EXP file, it is assumed by default that the file is a binary file. Otherwise, this must be changed in the TSK file (see [1], sections 2.2 and 3.2) by setting the following parameter:

```
with initial textual chrom. File   = 1      or in German
Mit initialem textuellem Chr.File  = 1
```

A chromosome file consists of one or more chromosomes. The files have the file extension MEM or AKS, which only affects the treatment when loaded by GLEAM:

- MEM: Loading all the chromosomes in the file into an empty chromosome storage. If the latter is not empty, it must first be cleared by dialog.
- AKS: loading all chromosomes of the file and extending the contents of the chromosome memory.

Files of both file formats can be concatenated. For example, a textual MEM file can be concatenated with a textual AKS file. Chromosome files should always be seen in conjunction with the corresponding gene model (MOD file, see also [2]). Attempting to interpret a chromosome in the context of an unsuitable gene model can lead to erroneous and surprising results!

If unsimulated chromosomes are loaded, their simulation and evaluation is performed so that they can be stored correctly in the chromosome memory and are available for initialization of an initial population[1].

Specifis from Version 2.2.2:

As of version 2.2.2 there is a new format for textual chromosome files, the difference being a more compact storage of genes. Therefore, the third section contains a revised version of the sub-sections 2.1.4 (Gene Elements) and 2.2 (Examples). The rest of the description of the section 2 is still valid.

In addition, the programs can be compiled with the activated switch `WITH_DYN_GENES`, which removes the restrictions on the size of a gene, i.e. the maximum number of integer and real parameters, as defined in the file `chaindef.h`. Please note that if `WITH_DYN_GENES` is set, only textual chromosome files can be loaded or written.

2 Structure of a Chromosome Text File up to (Hy)GLEAM Version 2.2.1

The text file contains one or more chromosomes in sequence, with line breaks between the elements of a chromosome and the chromosomes themselves. A chromosome consists of a *header element*, a *header parameter element*, one or more *segment descriptors* and one or more *gene elements*. It has the following structure:

```
+--------+--------+--------+     +--------+--------+     +--------+
| Hdr-   | Hdr-   | Segm-  | ... | Segm-  | Gene-  | ... | Gene-  |
| Elem   | Params | Descr 1|     | Descr m| Elem 1 |     | Elem n |
+--------+--------+--------+     +--------+--------+     +--------+
```

Each element is in one (long) line.

1: When using the external simulation services (ESS), this only works if individual simulations are possible and should therefore be tested at an early stage. E.g. by resetting the status field in the chromosome, which is easy to do with textual chromosome files.

2.1 Structure of the elements

The CONSTANTS and variables used here are contained in the file `chaindef.h` in the `sources` directory.

2.1.1 Header Element

Entries separated by at least one blank:

30200	fixed identifier (`activity`)
<fitness>	for unsimulated chromosomes 0 (`fitness_note`)
<refs>	number of references to this chromosome, 0 (`ref_counter`)
<laenge>	number of genes of the chromosome (`chain_length`)
<guete>	for unsimulated chromosomes 0 (`guete`)
<lfd-nr>	for unsimulated chromosomes 0 0 (`lfd_nr`)
<status>	status bits of the chromosome, 0 (`chain_status`)
<segm-anz>	number of segments of the chromosome (`segment_anz`)
<ak-idx>	internal management, can be 1 (`ak_index`)
12345	terminator

Status bits:

```
BASIS_AK                  1
TO_BE_SAVED               2
SIMULATED                 4
RANDOMLY_GENERATED        8
UNBEWERTETE_BASIS_AK  16384
UNMUTIERTE_URKETTE    32768
```

Note: Basic AKs are no longer relevant.

2.1.2 Header Parameter Element

This element contains the simulation results of the last simulation of the chromosome and consists of 16 (AK_ERG_WERTE_MAX) double values. For unsimulated chromosomes, this is 16 zeros.

2.1.3 Segment Descriptors

Segment descriptors contain the segment structure in the form of segment lengths. Each descriptor contains 12 (`SEGM_PTR_ANZ`) integer values. For the number of segments specified in the header, the segment descriptor elements must contain meaningful segment lengths. Meaningful means that they are each greater than zero and that their sum is equal to the chain length. If this requirement is not met, an error occurs during loading!

Note that the first value of the first Segment Descriptor element is unused and is represented by a zero. An incompletely used Segment Descriptor element is filled with zeros.

2.1.4 Gene Elements

A gene element consists of the gene type (`activity`), the gene parameters and the pointer identification. It contains the values of all possible gene parameters regardless of whether the gene model provides for them or not. Unused values are represented by zeros. Thus the parameter part of a gene element always consists of 12 (`I_PAR_ANZ_MAX`) integer values and 8 (`R_PAR_ANZ_MAX`) double values.
A gene element is terminated by its pointer identification. This is 12345 for all gene elements except the last gene element of a chromosome, which is terminated by 10000.

2.2 Examples

First a file with two chromosomes of the math. benchmark function “Foxholes”, where there are only two gene types, each with only one double parameter. The chromosome length is 2 each.

```
30200  0.000000000000e+00 0 2 0 0 0 2 0 12345               Chr1, header
  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0                        Chr1, header parameter
  0 1 1 0  0 0 0 0  0 0 0 0                                 Chr1, segment descriptor
1  0 0 0 0  0 0 0 0  0 0 0 0  0.5  0 0 0  0 0 0 0  12345    Chr1, 1st gene: gene type: 1
2  0 0 0 0  0 0 0 0  0 0 0 0  22.1 0 0 0  0 0 0 0  10000    Chr1, 2nd gene: gene type: 2
30200  0.0 0 2 0 0 0 1 0 12345                              Chr2, header
  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0                        Chr2, header parameter
  0 2 0 0  0 0 0 0  0 0 0 0                                 Chr2, segment descriptor
2  0 0 0 0  0 0 0 0  0 0 0 0   63.5 0 0 0  0 0 0 0  12345   Chr2, 1st gene: gene type: 2
1  0 0 0 0  0 0 0 0  0 0 0 0  -22.1 0 0 0  0 0 0 0  10000   Chr2, 2nd gene: gene type: 1
```

The first chromosome has two segments of length 1, the second only one segment of length 2. Both chromosomes have no address in the chromosome memory (<guete> and <lfd-nr> are both zero) and the status field is not set. The latter would lead to the fact that they are simulated and evaluated when they are read in at program start (specification of the file in the experiment file as value of the entry `Chromosome memory`), so that they can be sorted correctly into the memory. The following 2 indicates that the chromosome consists of two segments. The genes of the 2nd chromosome are in reverse order to the definition in the gene model.

Real numbers that are zero are written as 0. Otherwise in e-format with 12 decimal places. When reading in, of course, any valid format for double values is allowed according to the conventions of C.

The second example refers to the robot gene model (see gene model file `mitsu_hy_gb.mod` in the `InitFiles` directory), which contains chromosomes of dynamic length. An unsimulated and thus unevaluated chromosome of length 5 is shown. The simulated variant is then included in the second example of segment 3.2.

```
30200  0.000000000000e+00 1 5 1 1 8 2 1  12345
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 2 3 0 0 0 0 0 0 0 0 0
  1   0 0 0 0 0 0 0 0 0 0 0 0  7.20301924e-01 -7.7605766e+01 0 0 0 0 0 0  12345
  8   0 0 0 0 0 0 0 0 0 0 0 0  1.18683428e+00 0 0 0 0 0 0 0  12345
  2   0 0 0 0 0 0 0 0 0 0 0 0  7.35953664e+00  6.4200671e+00 0 0 0 0 0 0  12345
 10   0 0 0 0 0 0 0 0 0 0 0 0  2.87482336e+00 0 0 0 0 0 0 0  12345
 13  19 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0    10000
```

Fitness is zero, the chromosome consists of five genes and was located at address 1/1 in the chromosome memory. Bit 8, `RANDOMLY_GENERATED` is set in the status. The chromosome has two segments of length 2 and 3, as shown in the segment descriptor. Since the chromosome is unsimulated, all values of the header parameter element are zero. Gene types 1 and 2 have two real gene parameters and 8 and 10 have only one. The gene of type 13 consists of only one integer parameter.

3 Structure of a Chromosome Text File from (Hy)GLEAM Version 2.2.2

The differences between the two versions concern only the gene elements (section 2.1.4) and the example (section 2.2). Everything else said before is also valid in version 2.2.2 and following.

3.1 Gene Elements

A gene element consists of the gene type (`activity`), the gene parameters and the pointer identification. It contains the values of the gene parameters defined in the gene model. In contrast to earlier versions, there are no unused values which would be represented by zeros.

A gene element is terminated by its pointer identification. As before, this is 12345 for all gene elements except the last one, which is terminated with 10000.

3.2 Examples

First the representation of the same two chromosomes of the previous example for version 2.2.1 (section 2.2) in the new file format. As before, these are 2 chromosomes of the math. benchmark function "Foxholes", where there are only 2 gene types, each with only one double parameter. The chromosome length is 2 each.

```
30200  0.000000000000e+00 0 2 0 0 0 2 0 12345          Chr1, header
  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0                   Chr1, header parameter
  0 1 1 0  0 0 0 0  0 0 0 0                            Chr1, segment descriptor
1  5.000000000000e-01  12345                           Chr1, 1st gene: gene type: 1
2  2.210000000000e+01  10000                           Chr1, 2nd gene: gene type: 2
30200  0.0 0 2 0 0 0 1 0 12345                         Chr2, header
  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0                   Chr2, header parameter
  0 2 0 0  0 0 0 0  0 0 0 0                            Chr2, segment descriptor
2  6.350000000000e+01  12345                           Chr2, 1st gene: gene type: 2
1  -2.210000000000e+01  10000                          Chr2, 2nd gene: gene type: 1
```

Real numbers that are zero are written as 0. Otherwise in e format with 12 decimal places. When reading in, of course, any valid format for double values is allowed according to the conventions of C.

Genes without parameters consist only of the gene type (activity, here 12) and the pointer identification, e.g:

```
12      12345
```

The second example is the simulated variant of the second example of section 2.2, which differs in some header data and the header parameter element in addition to the abbreviated presentation of the genes.

```
   30200  1.048258848543e+01 1 5 2 1 12 2 1  12345
1.086721e+02 2.8000e+00 1.0000e+02 5.0000e+00 5.0000e+00 0 0 0 0 0 0 0 0 0 0 0
0 2 3 0 0 0 0 0 0 0 0 0
    1    7.203019237597e-01 -7.760576578970e+01   12345
    8    1.186834284136e+00   12345
    2    7.359536636808e+00 6.420067142906e+00   12345
   10    2.874823359411e+00   12345
   13  19     10000
```

According to simulation and evaluation, the chromosome has a fitness value of 10.48258848543 and was accordingly sorted under the address 2/1 when it was written into the file. The bit `SIMULATED` is now also set in the status. After simulation, the header parameter element now contains the values for the 6 criteria of this application.

4 Literature

[1] W. Jakob: *HyGLEAM - User Manual*. Technical Paper, KIT, IAI, 2020.
see `HyGLEAM-Manual_V1.0.pdf`

[2] W. Jakob: *MOD File Documentation*. Technical Paper, V1.3, IAI, 2020.
see `MOD-File-Docu_V1.3.pdf`