MOD File Documentation

GLEAM

Version 1.3

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

Release Notes

Changes to V1.1:

1. Editorial changes and clarifications
2. Some remarks on the maximum amount of integer and real parameters per gene type definition.

Changes to V1.2:

Description of the LHC configuration part.

1 Introduction

GLEAM distinguishes between a static and a dynamic interpretation of chromosomes. So far, the latter only plays a role in the coding of the application "collision-free robot path planning" (LESAK). Since GLEAM was created with this application, a large part of the documentation refers to the associated nomenclature. Therefore the following explanations:

1. The differentiation is only relevant for the interpretation of chromosomes.
2. A linguistic distinction is made between the gene and action model as follows:

static view on chromosomes:	dynamic or time related view on chromosomes:
gene model	action model (activity model)
chromosome	action chain (AC German: AK)
gene	action
parameter (decision variable)	parameter

A MOD file contains the gene or action model and consists of a descriptive header, rules for the construction of chromosomes (action chains) and the description of the genes (actions) and their parameters as well as their limits (explicit restrictions). This outlines an area of application in so far as it must be possible to describe the solution with the help of the gene model. The gene model is closely related to the simulator, which must interpret and "execute" the genes or actions in order to be able to evaluate the solution contained in the chromosome. An extended MOD file version for HyGLEAM also contains information on the local search methods or local hill climbers (LHCs, German: LSVs) and heuristics used. The GLEAM version is discussed first.

While the number of gene type definitions is not limited (apart from the available memory), the number of parameters per gene type is. The limits for integer and real parameters can be found in the `chaindef.h`-File: `I_PAR_ANZ_MAX` and `R_PAR_ANZ_MAX`. The actual values are 12 for the integers and 8 for the real parameters per gene type. From version 2.2.2 on these limitations can be overridden by setting the compiler switch `WITH_DYN_GENES` in the `schalter.h`-File. Since there is little experience with this extension, caution is advised and it is recommended to set the mentioned constants higher if problems occur.

The MOD file is an ASCII file and is created with a standard text editor.

2 MOD File Structure

2.1 MOD File Header

The header consists of at least 7 lines and has the following structure:

1st Line:
Comment line:

```
========= ... ======= Gene model for GLEAM/AE ======= ... ========
```

2nd Line:
MOD-file identification line:

```
GLEAM/AE <applicationClass> <application> [withLS]    [<timeStamp>]
```

with: `<applicationClass>` class of the application and the simulator interface
`<application>` name of the application or simulator. Must be present depending on the application class.

`withLS` optional indicator for the use of local searchers (LS). It indicates whether LHCs are to be used with HyGLEAM and the corresponding parameterization is contained in the MOD file.

Everything after the optional `withLS` identifier is skipped. The timestamp is for documentation purposes only. The following application classes are currently implemented, see also [1]:

`MathFkt` internal simulator. There are the following applications (benchmark functions):

`MBF-Ackley`	function according to Ackley
`MBF-Bigg`	function according to Bigg
`MBF-Fletcher`	function according to Fletcher and Powell
`MBF-Foxholes`	Shekel's foxholes
`MBF-Fractal`	fractal function
`MBF-Griewank`	function according to Griewank
`MBF-HValley`	helical valley
`MBF-Rast`	generalized Rastrigin function
`MBF-RBerg`	function according to Rechenberg
`MBF-Sphere`	sphere function
`MBF-WSphere`	weighted sphere function

`LESAK/Plus` internal robot simulator. No `application` specified.

`Matlab` Matlab simulator. The `application` is used as a switch for MatPower that indicates whether to calculate with OPF (`OPF`) or without (`PF`).

`ExtSimuServ` External Simulation Services (ESS). No `application` specified, as this is covered by the selection of a simulator model, which must be known and offered by the external simulation services.

`OPAL/V` OPAL/V Scheduling application for test purposes, no `application` specified

Old application class that is still known, but probably no longer usable:

`GADO-V2` with the simulators (`application`) `Mathematica`, Eldo and `GenExtSim` using the general interface for simulators based on pipes.

3rd Line:

The line consists of three numerical values followed by comments, which usually represent the variable names in the "hmod" package and indicate their permissible value ranges. These are generation specifications for the three chromosome types (the first two numbers) and for the segmentation (3rd number):

```
<gen_len_mode>  <gen_akt_mode>  <gen_segm_vert>
```

The first two numerical values determine the chromosome type as follows:

Chromosome with fixed length and irrelevant gene sequence (type 1):

`gen_len_mode`: 1 fixed length. The minimum chain length (line 4) determines the chromosome length.

`gen_act_mode`: 1 Each gene with a priority >0 is generated exactly once in ascending order.

Chromosome with fixed length and relevant gene sequence (type 2):

`gen_len_mode`: 1 fixed length. The minimum chain length (line 4) determines the chromosome length.

`gen_act_mode`: 2 Each gene with a priority >0 is generated exactly once in a random order.

Chromosome with dynamic length and relevant gene sequence (type 3):

`gen_len_mode`: 2 dynamic length. The generation length of the chromosome is within the range of the minimum and maximum chain length (line 4).

`gen_act_mode`: 3 For every gene type with a priority >0 a gene is generated in a random order according to the distribution formed by the priorities. The probability for generating a gene *j* is therefore

$$prio_j \;/\; \sum prio$$

This means that an action can occur not at all, once or several times.

The third numeric value, `gen_segm_vert`, is a specification for segmentation of newly generated chromosomes or for the resegmentation after a genetic repair, which may be necessary after crossover operations:

1 The segment length is determined equally distributed within the limits for segments defined in line 4. This is the default setting.

2 The segment length is determined with decreasing distribution within the limits for segments defined in line 4. For LESAK application or other applications with dynamic chromosomes.

4th Line:

The line consists of four numerical values and subsequent comments, which usually represent the variable names in the "hmod" package. These are generation specifications for the chain and segment length consisting of two values each:

`<min_ketten_len> <max_ketten_len> <min_abschn_len> <abschn_delta>`[1]

For chromosome types 1 and 2 the `min_ketten_len` is used. The `max_ketten_len` should be set to the same value and at least `min_ketten_len` gene types must be defined.
For chromosome type 3 must apply: `min_ketten_len` ≤ `max_ketten_len`.

A segment has at least the length `min_abschn_len` when generated. Therefore, this value must not be greater than `min_ketten_len.` `abschn_delta` specifies the maximum number of further actions (genes) a section may have.

5th Line:

The line indicates the number of gene types of the gene model. The following gene description part must contain the same number of gene type definitions in addition to the always existing definition of a sub-chain[2].

6th Line:

The line contains the number of result values of the simulator `akt_roh_erg_werte` and an optional control parameter for the genetic operators for small changes of gene parameters or decision variables resp. They are called `par_change_small` and `segm_change_small` in the EVO file. `<small_change_frac>` specifies the portion of the actual possible range of change to be used, which results from the current value of the decision variable and its value range limits. The default value is one thousandth.

`<akt_roh_erg_werte> [<small_change_frac>]`

When a MOD file is loaded, a field of length `akt_roh_erg_werte` is created, which is used to store the result values of a call to the (external) simulator and which is accessed by the fitness calculation. For this reason, the chromosome evaluation must not work with more evaluation criteria (given by the BEW file) than are specified by `akt_roh_erg_werte`. This is checked and, if negative, leads to termination for a CLV version of GLEAM. With TUI versions, error messages are displayed and the corresponding functions are locked.

1: ketten_len: length of a chain or chromosome. abschn: segment

2: Sub-chains no longer play a role

7th Line:

Number of possible application-specific additional files:

`<anw_file_anz>`

These files are introduced to the system below and are intended for simulator use. They are specified in the experiment file (EXP file).

Additional File Lines (starting from line 8):

The specification of additional files is omitted if `anw_file_anz` = 0. There is one line per additional file that contains the following:

`<ioCode> <ext> <fileDesignation> <menuEntry>`

`ioCode`:	0=read, 1=read/write, 2=write
`ext`:	file extension
`fileDesignation`:	File naming for dialogs (max. 15 characters)
`menuEntry`:	Entry in the appropriate menus (max. 10 characters)

2.2 LHC-Lines

This area is only available if the header (line 2) contains the identifier `witLS` (LS stands for local searcher, a synonym for local hill climber) Otherwise the area is omitted completely. If it exists, it is bracketed by two lines with the identifiers `BEGIN_LS` and `END_LS`. The configuration of local hill climbers (LHCs) or heuristics implemented in HyGLEAM or subsequently added to it requires knowledge of the adaptation smethod presented in [2]. The data of this section directly correspond to those treated by the “ShowEvoPar” and “Opt Params” menu items of HyGLEAM, with the scope depending on the currently selected default optimization method. For further details, see [3, Sect. 4.2.8].

For convenience, in this section local hill climbers and other heuristics are referred to as LHCs.

The following constants included in the file `evo.h` from package `evo` limit the number and properties of implemented LHCs:

- `LSV_ANZ` Number of implemented and managed LHCs
- `LSV_PAR_MAX` Maximum number of adaptively controlled LHC parameters
- `LSV_LEVEL_ANZ_MAX` Maximum number of levels per LHC parameter
- `LEVEL_MAX` Maximum number of levels for adaptive all-improvement per LHC

At present two LHCs are implemented, the Rosenbrock procedure with two (or three fin case of the `OPAL/V` application, see [1]) adaptively controlled parameters and the Complex algorithm with one. A short description of these LHCs and their control parameters can be found in [2].

The first two lines after Begin_LS indicate the number of active heuristics and their indices. In case of the standard implementation it looks like this

```
BEGIN_LS
 2  0            number of active LHCs   number of active heuristics
0  1             List of indices of active LHCs
```

Afterwards the default values of the LHC parameters follow in the order of the LHC indices, where `iteration limit` corresponds to $limit_R$ or $limit_C$ respectively and `abort limit` to th_R as described in [2]:

```
3000  0.02        LSV0 (Rosen): iteration limit  abort limit
300               LSV1 (Compl): iteration limit
```

These data configure the LHCs for their application without adaptation in the following optimization procedures: GLEAM with LHC improvement of a part of the initial population, pure LHC, simple MA (SMA).

The rest of the LHC part concerns the adaptation of LHC selection, some LHC parameters and the adaptive all-improvement. The levels and their start values are parameterized as well as the initial probability distribution of the LHCs involved. This is done separately per LHC in the order of their indices. As before, the description is illustrated by the standard configuration of the two implemented LHCs. Here `par0` corresponds to $limit_R$ or $limit_C$ respectively and `par1` to th_R.

First, the level probabilities per adapted parameter are determined, see [2, Table 1]:

```
10  100 200 350 500 750       1000 1250 1500 1750 2000         LSV0 (Rosen) par0
 9  .1 .01 .001 .0001 .00001  1.0e-06 1.0e-07 1.0e-08 1.0e-09  LSV0 (Rosen) par1
```

A level line start with the number of the levels involved followed by the level values of the corresponding adaptively controlled strategy parameter.

This is followed by the indices of the start level per parameter in one line and then the initial probabilities of the three active levels from the start level on. Only the first two initial probabilities of a level are given, because the third is the difference of the sum of the two to 1.

```
 0  0                  LSV0 (Rosen): startLevel of all parameters
 0.5 0.3  0.334 0.333  LSV0 (Rosen): initP1stLevel initP2ndLevel  of all parameters
```

The two lines are to be interpreted as follows: Both parameters start with the smallest level but with different level probabilities each. The first level of the iteration limit (`par0`) has an initial probability of 50%, the second of 30% and the third of 20%, while for the termination limit th_R (`par1`) all three levels have the same starting probability.

The description of a LHC is finalized by a line containing the initial probability of that LHC (`lsvStartP`) and initial values for the level adaptation of the adaptive all-improvement (starting at `allP`) of that LHC as before: first the index of the start level and then the initial probabilities of the three active levels from the start level on:

```
  0.5  0 0.334 0.333    LSV0: lsvStartP, allP: startLevel initP1stLevel initP2ndLevel
```

This is repeated for the Complex procedure , which has only one strategy parameter for adaptive adjustment:

```
10   50 100 150 200 300        400  500  650  800 1000         LSV1 (Compl) par0
  0                     LSV1 (Compl): startLevel of all parameters
  0.5 0.3               LSV1 (Compl): initP1stLevel initP2ndLevel  of all parameters
  0.5  0 0.334 0.333    LSV1: lsvStartP, allP: startLevel initP1stLevel initP2ndLevel
```

The last line contains the level values of the adaptive all-improvement which is the same for all involved LHCs. Note that the adaptation is done separately for each LHC and only the level values are the same.

```
 6  0.0 0.2 0.4 0.6 0.8 1.0  allP of all parameters
```

This means that the first level corresponds to the static best improvement, i.e. only the best offspring of a mating is improved locally. At the second level, additionally each sibling has an 20% probability of LHC improvement. At the second level it is 40% and so on.

As mentioned before the LHC section is finished by a line containing `END_LS.`

2.3 Gene Type Definitions

The area of the gene type definitions consists of 7 introductory comment lines, which describe the meaning of the entries of a gene type definition. Example in English:

```
*************************************************************************
 1) 2) 3)    4)     5)    1: priority of a gene    2: number of integer parameter
```

```
                          3: number of real param. 4: lower parameter limit
                          5: upper parameter limit
------------------------------------------------------------------------------
                          gene/parameter name   constant name    constant value
******************************************************************************
```

Example in German:

```
******************************************************************************
 1) 2) 3)    4)     5)      1: Prioritaet der Aktion  2: Anzahl Integer-Parameter
                            3: Anzahl Real-Parameter  4: Untere Parametergrenze
                            5: Obere Parametergrenze
------------------------------------------------------------------------------
                            Bezeichner/ParBez     Konstantennamen Konstantenwert
******************************************************************************
```

The definition of a gene type has the following structure:

```
<priority> <intParNo> <realParNo>       <geneName>
                           <ll>  <ul>     <paramName>       1st integer parameter
                                            <dimString>     1st integer parameter
                          .   .   .
                           <ll>  <ul>     <paramName>       nth integer parameter
                                            <dimString>     nth integer parameter
                           <ll>  <ul>     <paramName>       1st real parameter
                                            <dimString>     1st real parameter
                          .   .   .
                           <ll>  <ul>     <paramName>       mth real parameter
                                            <dimString>     mth real parameter
```

with: `<priority>` Integer. Probability portion of the gene during chromosome generation (0=off, for chromosome type 1 and 2 any number greater than 0 is sufficient for activation. Only for chromosome type 3, larger values in the sense of a probability distribution are useful. The probability of a gene *j* is equal to

$$\frac{priority_j}{\sum_{i=1}^{geneTypeNo} priority_i}$$

where *geneTypeNo* is the number of gene type definitions.

`<intParNo>` Number of integer parameters of the gene type (corresponds to above *n*)

`<realParNo>` Number of integer parameters of the gene type (corresponds to above *m*)

`<ll>` Integer or real. Lower limit of a gene parameter (decision variable)

`<ul>` Integer or real. Upper limit of a gene parameter (decision variable)

`<geneName>` Name of the action or gene for display purposes (maximum 20 characters (`ACT_NAME_MAX`))

`<paramName>` Name of the gene parameter for display purposes (maximum 20 characters (`PAR_NAME_MAX`))

`<dimString>` The dimension specification is for display purposes only and consists of three entries separated by blanks: The dimension identifier, a smaller/larger specification (<,>) and the alternative dimension identifier if the value is smaller or larger than 1/1000 or 1000 respectively (! ! ! = empty entry, the maximum length of a dimension identifier is 12 characters (`EINHEIT_MAX`)). The alternative dimension identifier is only used

when displaying simulation results, provided the application is configured to display the decision variables.

The description of the first gene type is fixed and consists of the following entry:

```
  0   0   0                                   sub-chain
```

3 Examples

The description of the structure of MOD files is illustrated by two examples. The first one is a simple gene model for the math. test function "Shekel's Foxholes" [1] consisting of 2 gene types, which are identical. The second example shows an action model for the LESAK application [1, 4, 5].

3.1 Simple Gene Model for the Benchmark Function "Shekel's Foxholes"

The actually read parts of a line are marked green.

```
************************** Gene model for GLEAM/AE ****************************
GLEAM/AE  MathFkt  MBF-Foxholes                                          5.05.1998
1  1  1           gen_len_mode (1,2)  gen_akt_mode (1,2,3)  gen_segm_vert (1,2)
2  2    1   0     min_ketten_len  max_ketten_len  min_abschn_len  abschn_delta
 2               number of gene types
 1               akt_roh_erg_werte of (ext.)simu
 0               number of additional files
*******************************************************************************
 1) 2) 3)    4)     5)    1: priority of a gene    2: number of integer parameter
                          3: number of real param. 4: lower parameter limit
                          5: upper parameter limit
-------------------------------------------------------------------------------
                          gene/parameter name   constant name    constant value
*******************************************************************************
  0  0  0                   sub-chain             ACTIVITY_ACT        (00)
  1  0  1                   par1
            -500.0  +500.0     p1
                               ! ! !
  1  0  1                   par2
            -500.0  +500.0     p2
                               ! ! !
```

3.2 Action model of the LESAK application

The actually read parts of a line are marked green.

```
************************ Action Model for GLEAM/AE ****************************
GLEAM/AE   LESAK/Plus                              01.07.1998   revised: 29.06.2020
2  3  2           gen_len_mode (1,2)  gen_akt_mode (1,2,3)  gen_segm_vert (1,2)
5  50   4   6     min_ketten_len  max_ketten_len  min_abschn_len  abschn_delta
13               number of action types (gene types)
 7   0.002       akt_roh_erg_werte of (ext.)simu   small_change_frac (optional)
 2               number of additional files
0 kin Kinematics-File Kinematics
0 obs Obstacle-File Obstacles
*******************************************************************************
 1) 2) 3)    4)     5)    1: priority of an action 2: number of integer parameter
                          3: number of real param. 4: lower parameter limit
                          5: upper parameter limit maxNameLength: gene=20 param=10
-------------------------------------------------------------------------------
                          gene/parameter name   constant name    constant value
*******************************************************************************
  0  0  0                   sub-chain             ACTIVITY_ACT        (00)
```

```
  5   0   2                  SetMotorSpeed_M1      MOTOR_1              (01)
                0.1   15.0       ramp
                                    ! ! !
              -120.0  120.0       voltVal_1
                                    V < mV
  5   0   2                  SetMotorSpeed_M2      MOTOR_2              (02)
                0.1   15.0       ramp
                                    ! ! !
              -120.0  120.0       voltVal_2
                                    V < mV
  5   0   2                  SetMotorSpeed_M3      MOTOR_3              (03)
                0.1   15.0       ramp
                                    ! ! !
              -120.0  120.0       voltVal_3
                                    V < mV
  5   0   2                  SetMotorSpeed_M4      MOTOR_4              (04)
                0.1   15.0       ramp
                                    ! ! !
              -120.0  120.0       voltVal_4
                                    V < mV
  5   0   2                  SetMotorSpeed_M5      MOTOR_5              (05)
                0.1   15.0       ramp
                                    ! ! !
              -120.0  120.0       voltVal_5
                                    V < mV
  5   0   1                  Motor1_off            MOTOR_1_AUS          (06)
                0.1   12.0       ramp
                                    ! ! !
  5   0   1                  Motor2_off            MOTOR_2_AUS          (07)
                0.1   12.0       ramp
                                    ! ! !
  5   0   1                  Motor3_off            MOTOR_3_AUS          (08)
                0.1   12.0       ramp
                                    ! ! !
  5   0   1                  Motor4_off            MOTOR_4_AUS          (09)
                0.1   12.0       ramp
                                    ! ! !
  5   0   1                  Motor5_off            MOTOR_5_AUS          (10)
                0.1   12.0       ramp
                                    ! ! !
 10   0   0                  BlockBegin            START_BLOCK_BEGIN    (11)
 10   0   0                  BlockEnd              START_BLOCK_END      (12)
 30   1   0                  Unchanged             UNVERAENDERT         (13)
                1     20         cycles
                                    ! ! !
```

The two columns *constant name* and *constant value* are for documentation purposes only.

For each motor of the 5-axis industrial robot, individual actions (genes) are defined for moving and switching off. This is followed by the three actions of time control, see also [4, section 4.2] or [5]

Instead of defining separate actions for each motor, the motor number could have been made a further parameter of a general action for starting and stopping a motor movement respectively. This would have simplified the action model. However, this was not done, because otherwise the new parameter *motor number* would have acquired a great phenotypic significance and a considerable disproportion would have arisen between the phenotypic effect of a small change in this parameter and an equally small change in one of the other gene parameters. This would have violated the requirement for the strongest possible causality in the definition of the gene model. In fact, it could also be shown that the action model presented here is superior to one consisting of two universal movement actions with motor numbers as parameters.

4 Literature

[1] W. Jakob: *Applications Included in GLEAM and Integration of New Ones.* Technical Paper, V1.0, IAI, 2020.
See `GLEAM-Applications_Docu_V1.0.pdf`

[2] W. Jakob: *A general cost-benefit-based adaptation framework for multimeme algorithms.* Memetic Computing, 2(2010) 201-18 doi: 10.1007/s12293-010-0040-9
A preprint is included in the literature sub directory of the documentation directory.

[3] W. Jakob: *HyGLEAM - Hybrid General Purpose Evolutionary Algorithm and Method: User Manual.* Technical Paper, V1.0, IAI, 2020.
See `HyGLEAM-Manual_V1.0.pdf`

[4] C. Blume, W. Jakob: *GLEAM - General Learning Evolutionary Algorithm and Method: Ein Evolutionärer Algorithmus und seine Anwendungen*. (in German) Karlsruhe: KIT Scientific Publishing, 2009. doi: 10.5445/KSP/1000013553

[5] C. Blume, W. Jakob: *GLEAM - An Evolutionary Algorithm for Planning and Control Based on Evolution Strategy*. In: E. Cantù-Paz (ed.): Conf. Proc. of Genetic and Evolutionary Computation Conference (GECCO 2002), New York, Vol. Late Breaking Papers (LBP), 2002, pp.31-38