HyGLEAM

Hybrid General Purpose Evolutionary Algorithm and Method

User Manual

Version 1.0

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 V1.0:
1.

1 Introduction

A guide to the successful application of Evolutionary Algorithms (EA) in general can be found in [1]. This guide is especially intended for inexperienced users with basic knowledge about EAs.

The software package allows a separate generation of GLEAM or HyGLEAM [2, 3]. GLEAM consists of the evolutionary algorithm (EA) of the same name as a stand alone procedure. HyGLEAM contains additionally local search procedures, at present the Rosenbrock procedure and the Complex algorithm. Interfaces are provided to add further heuristics as local searchers. HyGLEAM allows the execution of all optimizers separately as well as in conjunction as so called *Memetic Algorithms* (MA). For the latter three different optimizers are implemented as described below. Additionally, the initial populations of GLEAM and the MAs can bee seeded using the local searchers and/or some loaded solutions generated externally or resulting from previous runs.

From both programs two versions exist: a command line version (CLV) and a version with a textual user interface (TUI) based on simple menus running in any shell, see also [4]. The implementation of a more user-friendly GUI version had to be canceled due to time constraints.

There are English and German variants of all GLEAM and HyGLEAM versions available. The nomenclature of the English variants is more strongly influenced by the genetic model, while the German variants are dominated by the action model, see also [2] and there Sect. 4.1 in particular. The following table summarizes this:

Gene Model	Action Model	Aktionsmodell
chromosome	action chain	Aktionskette (AK)
chromosome type	type of action chain	Aktionskettentyp (AK-Typ)
gene type	action type	Aktionstyp
gene	action	Aktion
decision variable	action parameter	Genparameter or Parameter
fitness	fitness	Note or Fitness
fitness class	fitness class	Güte or Güteklasse
chromosome address: class/index	chain address: class/index	AK-Adresse: Güte/Lfd.Nr

In terms of content, there are no differences in chromosomes between the two models [3]. The differences concern the interpretation: The gene model is interpreted statically and without time reference, while the action model is interpreted time-related and dynamically, see also [2, Sect. 4.2, 3].

Installation

There is no installation tool available and necessary. The program file may reside anywhere in the file system and an alias name may be used to call it. Or it may be called by providing the necessary path. The program expects a file containing the error messages to reside in the working directory or in a directory, the environment variable `GLEAM_ROOT` points at, see also the beginning of Sect. 3. The use of a directory that is used (only) for the error message file and the (standard) initialization files of GLEAM is strongly recommended. Experience shows that some variants of the initialization files are quickly created and erased, which is easier to manage separately in the working directory.

The distribution contains executable programs of the English TUI and CLV variants compiled for Ubuntu in the `linux` directory. The `gleam_ae` sub directory also contains a German-language TUI variant.

The recommended directory structure is described in [4]. The files of the distribution are organized and stored accordingly and all outputs and messages discussed in this documentation are based on

this and the following path of the `gleam` directory, which has to be adjusted accordingly during an installation:

```
/home/wilfried/gleam/
```

The first corresponding adjustment concerns the declaration of the environment variable GLEAM_ROOT, which must read based on the above path:

```
GLEAM_ROOT=/home/wilfried/gleam/InitFiles
```

And if compilation is to be done, the `glob_inc.mak` files are also affected, see also [4, Sect. 3]. The declaration of `ROOT` at the beginning of these files would then have to be adapted and currently reads:

```
ROOT = /home/wilfried/gleam
```

Implemented Applications

There are two applications integrated in GLEAM and HyGLEAM [5], a set of mathematical benchmark functions (MBF, `MathFkt`) [5] and a collision-free path planning task for industrial robots (`LESAK/Plus`) [2, 5, 6]. The configuration of an application is described in [7]. The application class IDs used in some initialization files are given in brackets, see also Sect. 3.1 and 3.2 and [7].
There is another application regarding a scheduling task from process industry (`OPAL/V`) [2, 8], which serves as an example for the integration of a new application, see [5]. As with all other new applications this one is integrated in addition to the two standard (test) applications.

Terminology

Since an EA or an MA is closely linked to an application, especially by gene model and evaluation, we speak of a GLEAM or HyGLEAM application, or an **application** for short. An application can be represented by at least two program **versions**: the CLV and the TUI version. Of these, linguistic **variants** can exist, currently for English and German. The latter has influence on the initialization files which contain name-value pairs, i.e. on the EXP and TSK files. In this sense, the distribution contains three applications in their own application directories, namely:

- GLEAM/AE in the `gleam_ae` directory containing the two applications mentioned above.
- HyGLEAM/A in the `hy_gleam_a` directory containing the same two applications.
- HyGLEAM/OPALV in the `hy_gleam_opalv` directory containing the above mentioned OPAL/V application in addition to the two standard applications.

2 Command Line Version

There is one common main program for both the command line version of GLEAM and HyGLEAM. The difference is that for GLEAM no local hill climbers or other heuristics are available. These additional procedures are summarized under the abbreviation LHC for local hill climber, sometimes also referred to as local searchers (LS). The rest of the behavior of both versions is the same. The command line version allows

- the specification of one optimization job by command line parameters and by parameters given in an initialization file called task file (.tsk),
- the optional initialization of the initial population from chromosomes stored in a file (.mem),
- the execution of an optimization job,
- the generation of a log file containing information of the performed optimization run and error messages, if any, and
- the optional simulation of the result and its further processing.

The HyGLEAM engine can perform the following optimization job types:

1. GLEAM: Execution of the evolutionary algorithm GLEAM, whereby a part of the starting population can be improved by means of a selected LHC (pre-optimization)
2. LHC: Performs one of the local hill climbers (LHCs)
3. SMA: Simple Memetic Algorithm (GLEAM plus one LHC)
4. ASMA: Adaptive Simple Memetic Algorithm (GLEAM plus one LHC whose strategy parameters are adapted)
5. AMMA: Adaptive multimeme algorithm (GLEAM plus several LHCs). Apart from the strategy parameters of the LHCs, their application probability as well as the deactivation of poor performing LHCs are adapted. In addition, with *all-improvement*, the likelihood of having siblings of the best offspring of a mating locally improved is also adaptively adjusted.

The command line version requires the specification of an existing experiment file (.exp) which summarizes all other initialization files required for the experiment at hand. The contents of this file is the same as with the interactive version, see Sect. 3.

2.1 Program Call

The program of the standard HyGLEAM command line version is called as follows, while variants may have different program names usually ending with CLV:

hyGleamCLV [OPTIONS]... EXP_FILE [LOG_FILE]

with

EXP_FILE: filename of the experiment file (with path). Lists all files involved.

LOG_FILE: filename of the log file (with path). If not specified, the name of the experiment file is used.

-A<text> : acceptance rule for offspring: *always, localLeast, betterParent, always-ES, localLeast-ES*

-B(+|-) : +: best-improvement, -: all-improvement

-C(+|-) : With/without creation of a case file when simulating the final result. Only for MATPOWER applications!

-D<demeSize> : neighbourhood size, must be even and smaller than popSize. (4 - 32)

-E<number> : limit for function evaluations (0 or not specified: termination condition switched off)

-F<fitness> : fitness limit (termination condition) (0.0 - 100,000.0)

-G<gen> : generation limit (termination condition)

-h -H –help : online help (this text)

-I<text> : initialization strategy: *new, best, mix, bestNew, lhcNew, lhcBest*

-i<value> : initialization parameter: mix: minimal fitness, bestNew: amount of best, lhcNew: amount of LHC improved chromosomes, lhcBest: amount of best

-L(+|-) : Lamarckian evolution: +:on, -:off (Baldwinian evolution)

-l<frac> : learning rate: proportion of old LHC distribution used for the new one in % (0.0 - 100.0)

-M<value> : maximum iterations per LHC run (termination threshold)

-N<domain> : domain name used for the external simulations service (ESS)

-n<slaveNo> : 0: master, >0: slave number, used by PAR_POP program versions

-O<text> : optimization procedure: *gleam, sma, asma, amma, lhc*

-P<popSize> : population size (4 - 20,000)

-p(+|-) : logging: +:detailled, -:minimal

-R<value> : ranking parameter, from minimal to maximal selective pressure (1.0 - 2.0)

-r<number> : number of results. Default: 1, max: popSize/2, 0: no simulation or save of results possible.

-S<index> : Searcher: LHC index for *lhc* jobs or *gleam, sma, asma,* or *amma* jobs with LHC-initialization (*lhcNew* or *lhcBest*)

-T<value> : time limit: maximal job run time in seconds (termination condition)

-t<value> : threshold for LHCs (optional 2nd LHC termination threshold)

-W<nr>:<val> : new weigt <val> of criterion <nr>. The weight must be given as maximum fitness value (0 .. 100,00) and <nr> must be a valid number of a criterion.

-X : reserved for undocumented experimental settings

If only the capital letter is specified for an option, the lowercase letter also applies.

The options are explained along with their associated TSK file parameters in detail in Sect. 2.2.

All provided options override the corresponding task file parameters without notification. Only if an option specifies a different optimization procedure than the one given in the task file, a warning is issued. Options or task file parameters, which are not required for the selected optimization procedure, are ignored. Erroneous specifications of options like a faulty number (e.g. -Mx4z or -M102.5) are detected only for options, which are relevant for the actual optimization procedure or GLEAM application. If a relevant option is faulty, it is usually considered a fatal error, which leads to program termination. The rationale for this is that the user's intention cannot be considered correctly and that thus, the resulting run based on default values is probably not intended.

In the end, all necessary job parameters must be provided, either by command line options, or by parameter settings of the task file, or by program default values, and they must be consistent. E.g. the size of the neighborhood (deme size) must be even and it must not exceed the population size.

2.2 Task File Parameters

Table 1 lists the task file parameters, their data types, default values, whether they are required in which context, and their availability as a command line option. Most of the parameters are intended to control the command line version. For interactive versions they are frequently not needed and can serve as a different setting of default values, which may be changed by the user through the appropriate menus or when defining an optimization job. Thus, it is possible to use a TSK file which was written for controlling runs of the command line version for the interactive versions as well.

As with command line options, the specifications of a parameter may be faulty due to invalid numbers, unknown enumeration values or the like. Such a case is treated as a fatal error, the processing of the file is stopped and the program is terminated immediately. This also applies to parameters, which are not required for the selected optimization procedure, as this may be either not known when reading the erroneous parameter or the optimization procedure may be selected or altered later. Consequently, task files may contain parameters specified correctly but not needed in the actual optimization context. But they must not contain faulty parameter specifications regardless of their need.

Boolean parameters can take the values of "yes" or 1 and "no" or 0 respectively. For the German variant "ja" and "nein" apply.

The table consists of four parts: the general parameters of all GLEAM applications and versions, the HyGLEAM-specific ones, application dependent parameters, and parameters of the command line version only. There is a special data type, enumeration (*enum.* in the table), which is translated to integer.

Denotation	Type	Default	Required	Edit.	CLV-Option
with init. of random generator	bool	yes	optional	yes	no
with recording of best of evolution	bool	yes	optional	yes	no
with recording of statistical data	bool	yes	optional	yes	no
with stat.data recording per gen.	bool	no	optional	yes	no
with initial textual chrom. file	bool	no	optional	no	no
optimization procedure	enum.	GLEAM	optional	optP	-(O|o)<text>
acceptance rule for offspring	enum.	LLES	optional	optP	-(A|a)<text>
minimal hamming dist.f.XO/Reco [%]	double	0.1	optional	optP	no
ranking parameter	double	1.4	optional	optP	-R<value>
population size	integer	120	optional	yes	-P<size>
deme size	integer	8	optional	optP	-(D|d)<size>
init. strategy for start population	enum.	new	optional	yes	-I<text>
init. strategy parameter	integer	0/1 *)	required for CLV if no -i-option	yes	-i<value>
number of results	integer	1	optional	yes	-r<number>
target fitness	double	100000	optional	yes	-(F|f)<fitness>
limit for generations	integer	1000	optional	yes	-(G|g)<gen>
max.gen. w/o child accept. in deme	integer	100	optional	yes	no
max.gen. w/o deme improvement (GDI)	integer	400	optional	yes	no
limit for evaluations	integer	INT_MAX	optional	yes	-(E|e)<limit>
run time limit in seconds	integer	3600	optional	yes	-T<sec>
backup rate [generations]	integer	INT_MAX	optional	yes	no
number of digits of (simu)parameter	integer	7	optional	yes	no
MA with all-improvement	bool	no	optional	(yes)	-(B|b)(+|-)
LHC (Local Hill Climber) index	integer	0	optional	(optP)	-(S|s)<index>
Lamarck rate [%]	double	100.0	optional	(optP)	-L(+|-)
LHC iteration limit	integer	**)	requ. according to optim., if no -M	(optP)	-(M|m)<limit>
LHC termination threshold	double	**/	requ. according to optim., if no -t	(optP)	-t<value>
speed of LHC adaption	enum.	fast	optional, used for AMMA only	(optP)	no
speed of level adaption	enum.	fast	optional, used for adaptive MAs	(optP)	no
number of fitness classes	integer	3	optional, used for adaptive MAs	(optP)	no
fraction of old LHC-distribution[%]	double	33.333	optional, used by AMMA only	(optP)	-I<fraction>
rotation angle in degrees	double	0	optional, for math. test functions	(yes)	no
with phenotypic repair	bool	(no)	optional, default depends on impl.	(optP)	no
genetic repair rate [%]	double	(0.0)	optional, default depends on impl.	(optP)	no
simulator model	string	-	extSim: optional	(yes)	no
timeout for ext. simulator [sec]	integer	-	extSim: optional	(yes)	no
runs per (re)init. of ext.simu.	integer	10000000	extSim: optional	(yes)	no
with logging of ext.simu. interface	bool	no	extSim: optional	(simu)	no
with display of ext.simu. runtime	bool	no	extSim: optional	(yes)	no
with Mitsubishi R500	bool	yes	optional, robot appl. LESAK/PLus	(yes)	no
with collision test	bool	yes	optional, robot appl. LESAK/PLus	(yes)	no
with end stop test	bool	yes	optional, robot appl. LESAK/PLus	(yes)	no
with depreciation on stop/crash	bool	yes	optional, robot appl. LESAK/PLus	(yes)	no
number of axes	integer	5	optional, robot appl. LESAK/PLus	(yes)	no
simulator cycle time [sec]	double	0.1	optional, robot appl. LESAK/PLus	(yes)	no
CEC: limit of fct.evaluations (FEs)	integer	-	optional, requ. for a CEC benchm.	(yes)	no
Domain name of extern. SimuServices	string	***)	optional, requ. for ext.Sim.Serv.	(yes)	-N<domain>
Max. employee peak per shift	double	9.0	optional, required for OPAL-appl.	(yes)	no
with detailed logging	bool	yes	optional	no	-p(+|-)
with result simulation	bool	no	optional, requires at least 1 result	no	no
with saving of result chromosomes	bool	no	optional, requires at least 1 result	no	no
with job termination tests	bool	no	optional	no	no
termination test rate [pairings]	integer	500	optional	no	no

Table 1: TSK file parameters and some of their properties. For the meaning of the colors see the text above. The "Edit."-column indicates the editability and is explained in Sect. 4.1. Abbreviations: appl.: application,

CLV: Command Line Version, enum.: enumeration, impl.: implementation, init.: initialization, math.: mathematical, optim.: optimization procedure, popul.: population, requ.: required

*: The default value depends on the initialization procedure selected.

**: The default values of the two HyGLEAM parameters for LHC control can not be included in the table, as they are taken from the MOD file, see Sect. 3.1.

***: The default value is something like "undefined.iai.kit.edu".

Short description of the task file parameters:

The parameters are listed in the sequence of Table 1. The description is followed by a line with the constant name and index of the parameter identifier, the associated variable and the package and module of its declaration and initialization. If the modules of declaration and initialization or setting are different, the latter is also given. If this is done by a function different from the package initialization routines, this function is given in brackets.

General parameters of all GLEAM applications and versions:

with init. of random generator **Mit Init. d.Zufallszahlengenerators**

Controls the initialization of the random generator. Usually switched off for debugging purposes only.
WITH_RANDGEN_INIT_TXT/_IDX, initZufGen, evo: evo_gsw.c

with recording of best of evolution **Merke Daten des Evolutionsbesten**

During the course of evolution, the data of the currently best individual is recorded. Useful for display purposes of interactive versions.
WITH_REC_EVO_BEST_TXT/_IDX, record_evo_best, bew: bewert.c

with recording of statistical data **Mit Statistikdaten-Sammlung**

Controls the collection of statistical data of an optimization job.
WITH_STAT_DAT_COL_TXT/_IDX, mit_statistik, evo: evo_steu.c

with stat.data recording per gen. **Mit StatDatenSammlung je Generation**

Controls the collection of statistical data per generation of an evolution based optimization job. Requires the activation of standard recording per job and is much more detailed. May slow down the optimization a little bit.
WITH_GEN_STAT_COL_TXT/_IDX, statistik_pro_gen, simu: simu.c

with initial textual chrom. file **Mit initialem textuellem Chr.File**

Controls the type of the optional initial chromosome file, which can be either binary (default) or textual.
WITH_GEN_STAT_COL_TXT/_IDX, initialChrFileMode, chio: chain_io.c

optimization procedure **Optimierungsverfahren**

Supported are the following optimization procedures: *gleam, sma, asma, amma,* and *lhc,* the names not being case sensitive. In Geman: g*leam, sma, asma, amma,* und *lsv.*
GLEAM: Optional and must be *gleam,* if given.
HyGLEAM: Optional, all procedures other than GLEAM require the availability of active LHCs.
OPT_PROCEDURE_TXT/_IDX, def_opt_strat, evo: evo_gsw.c

acceptance rule for offspring **Akzeptanzregel fuer Nachkommen**

Rule for the acceptance of the best offspring per pairing with respect to the deme of its parent. Elitist rule variants (ES) also require that the best individual of the neighborhood be replaced only if the best offspring itself is better. Valid rules with their acronyms for job display purposes in brackets:

always: The best offspring always replaces its parent. (Alw)
localLeast: The best offspring replaces its parent only if it is better than the weakest individual of the local deme, i.e. neighborhood. (LL)
always-ES: Elitist variant of *always*. (AlwES)
localLeast-ES: Elitist variant of *localLeast*. (LLES)

betterParent: The best offspring replaces its parent only if it is better than the parent. (BP)

The same identifiers are used in German.
ACCEPTANCE_RULE_TXT/_IDX, default_survival_rule, evo: evo_gsw.c

minimal hamming dist.f.XO/Reco [%] Mindesthammingabstand f.XO/Reko [%]

Threshold value for the hamming distance of two chromosomes in percent above which the cross-over and recombination operators [2, 3] are performed. This is intended to avoid cross-over or recombination of too similar individuals. Valid range: 0.0 (perform XO/Reco always except for identical parents) – 100.0 (do not perform XO/Reco at all)
XO_HAMDIST_TXT/_IDX, def_xo_min_ham_dist, evo: evo_gsw.c

ranking parameter Ranking-Parameter

This parameter controls linear ranking for partner selection and by that also selective pressure. The minimum value of 1.0 means random selection (i.e. no selective pressure when choosing a mate), while the maximum value of 2.0 corresponds to the highest selective pressure possible with linear ranking.
RANKING_PAR_TXT/_IDX, def_max_fit, evo: evo_gsw.c

population size Populationsgroesse

The population size is required for every population based optimizer. Thus, it can be omitted for LHC only. Range: 4 – 20000. The present range of an implementation can be obtained by calling the program with -H or –help.
POP_SIZE_TXT/_IDX, def_psize, evo: evo_gsw.c

deme size Nachbarschaftsgroesse

Size of the neighborhood of an individual (deme). It is required for every population based optimizer and must be even and not larger than the population size. Sizes range from 4 to 32, see also the -H or --help option. It is recommended that deme sizes are (much) smaller than the population size. Typical values are 8 or somewhat more for larger populations of more than several hundred individuals.
DEME_SIZE_TXT/_IDX, anz_nachbarn, evo: evo_gsw.c

init. strategy for start population InitStrategie f.die Startpopulation

Method to construct the initial population for a population based optimization procedure or to select an initial solution for an LHC optimization. If chromosomes are to be chosen from the chromosome memory this is always done under avoidance of multiple selection.

GLEAM:

new: Create all individuals randomly and new. In German: *neu*

best: Select the best μ individuals from chromosome memory, where μ is the population size. If there are not enough available, the rest will be randomly generated as with *new*. In German: *best*

mix: Select μ individuals from memory with a fitness better than *f*, where μ is the population size and *f* is given by the *initStrategy parameter* (default: 0). If there are not enough individuals available, the rest will be randomly generated as with *new*. In German: *mix*

bestNew: Select the best *n* individuals from chromosome memory, where *n* is given by the *initStrategy parameter* (default: 1) and must not be larger than the population size μ. The rest will be generated randomly as with *new*. In German: *bestNeu*

HyGLEAM: additional strategies for population based optimizers, whereby the LHC as given by the *LS procedure* is used:

lhcNew: All individuals are generated randomly and *n* of them are locally optimized, where *n* is given by the *initStrategy parameter* (default: 1) and must not be larger than the population size μ. In German: *lsvNeu*

lhcBest: Create the initial population according to *bestNew* and optimize all locally. In German: *lsvBest*

Strategies for LHC: Only one individual is selected by strategy *new* or *best*.
INIT_STRATEGY_TXT/_IDX, defInitStrat, evo: evo_gsw.c

init. strategy parameter **InitStrategie-Parameter**

Required for initialization strategies *mix*, *bestNew*, *lhcNew*, and *lhcBest*. For the meaning see initialization strategies. This parameter or its corresponding option is required for the command line versions.
INIT_SPAR_TXT/_IDX, defInitStratPar, evo: evo_gsw.c

number of results **Anzahl der Ergebnis-Chromosome**

Number of result chromosomes, which will be extracted from the population after job termination. It is intended to extract individuals with good fitness values, which differ as much as possible. The number is limited to the half of the population size and reduced if necessary.
NUMBER_OF_RESULTS_TXT/_IDX, defResultChrs, evo: evo_gsw.c

target fitness **Mindest-Zielfitness**

This termination criterion specifies a fitness value between 0.0 (poor) and 100000.0 (best) at which the optimization run is terminated. If the initial population already contains an individual of this quality or better the run is not started at all. This also holds for the start individual in case of a LHC run.
TARGET_FITNESS_TXT/_IDX, defTargetFitness, evo: evo_gsw.c

limit for generations **Maximale Generationsanzahl**

Maximum number of generations of a job. (termination criterion)
GEN_LIMIT_TXT/_IDX, defGenLimit, evo: evo_gsw.c

max.gen. w/o child accept. in deme **Max. Gen. ohne Akzeptanz im Deme**

Maximum number of generations without any acceptance of an offspring in all demes. (termination criterion)
GAC_LIMIT_TXT/_IDX, defGAcLimit, evo: evo_gsw.c

max.gen. w/o deme improvement (GDI) **Max. Gen. ohne Deme-Verbesserung**

Maximum number of generations without an improvement of the best individual of any deme. (termination criterion)
GDI_LIMIT_TXT/_IDX, defGDI_Limit, evo: evo_gsw.c

limit for evaluations **Maximale Anzahl der Evaluationen**

Maximum number of fitness function evaluations. This additional termination criterion can be used for comparisons between different algorithms. The default is INT_MAX and thus, it is deactivated.
EVAL_LIMIT_TXT/_IDX, defEvalLimit, evo: evo_gsw.c

run time limit in seconds **Maximale Laufzeit [sec]**

Maximum run time of a job in seconds. (termination criterion)
RUNTIME_LIMIT_TXT/_IDX, defTimeLimit, evo: evo_gsw.c

backup rate [generations] **Backup rate [Generationen]**

Specifies the number of generations, after which a copy of the population is written to a file named "evo_tmp.mem". This is intended to save the latest state of the population for further use by an interactive version of GLEAM or HyGLEAM. The default value is INT_MAX, which means that no backups are made.
BACKUP_RATE_TXT/_IDX, save_rate_def, evo: evo_gsw.c

number of digits of (simu)parameter **Stellenanzahl der (Simu)-Parameter**

This parameter controls the number of digits for converting double values to strings. These strings are used for display purposes, for documentation (.res file), and may also be used for communication with an external simulator.
SIMU_RES_PREC_TXT/_IDX, erg_genauigkeit, simu: simu.c

HyGLEAM-specific parameters:

MA with all-improvement **MA mit all-Verbesserung**

Optional parameter for the three MAs of HyGLEAM. In case of the SMA all offspring of a pairing are improved by the LHC. In case of the two adaptive MAs ASMA and AMMA, the

best offspring of a pairing and an adaptively controlled fraction of its siblings undergo LHC improvement.
MA_WITH_ALL_IMPR_TXT/_IDX, withAllImprovement, evo: evo_gsw.c

LHC (Local Hill Climber) index **LSV-Index**

Index of one of the implemented and activated local searchers or local hill climbers (LHC). Standard HyGLEAM contains the Rosenbrock procedure (index = 0) and the Complex algorithm (index = 1). Required for HyGLEAM only if initialization of the start population involves local improvement or for the following optimizers: SMA, ASMA and LHC.
LHC_INDEX_TXT/_IDX, def_lsv_strat, evo: evo_gsw.c

Lamarck rate [%] **Lamarck-Rate [%]**

The Lamarck rate controls the probability of choosing Lamarckian evolution, in which case the genes of a chromosome are updated according to the LHC result. Otherwise, the improved fitness is used only and the chromosome remains unchanged (Baldwinian evolution). As experience showed, the combination of structured populations and Lamarckian evolution works usually very well. Thus, the command line option allows the choice between both alternatives only, resulting in a value of 100 % or 0 %, respectively.
LAMARCK_RATE_TXT/_IDX, def_max_fit, evo: evo_gsw.c

LHC iteration limit **LSV-Iterationslimit**

Termination criterion for iterative LHCs. Default values come from the gene model file (MOD file). Remark: This parameter is also considered in the interactive versions of GLEAM.
LHC_ITER_LIMIT_TXT/_IDX, lsv[].lsvPar[0], evo: lsv_steu.c

LHC termination threshold **LSV-Abbruchschranke**

This additional termination criterion is used for LHCs like the Rosenbrock procedure, in which case it has a range between 0.0 (no stopping) and 1.0 (immediate stop). In general, its meaning depends on the used LHC. Default values come from the gene model file (MOD file). This parameter is also considered in the interactive versions of GLEAM.
LHC_TERM_LIMIT_TXT/_IDX, lsv[].lsvPar[1], evo: lsv_steu.c

speed of LHC adaption **LSV-Adaptionsgeschwindigkeit**

There are three predefined adaptation speeds available for the adaptation of the LHC distribution: *fast*, *medium*, and *slow*. In German: *langsam*, *mittel*, *schnell*
LHC_ADAPT_SPEED_TXT/_IDX, -, evo: adapt_di.c (setLSVAdaptSpeed())

speed of level adaption **Level-Adaptionsgeschwindigkeit**

There are three predefined adaptation speeds available for the adaptation of the parameter levels of each LHC: *fast*, *medium*, and *slow*. In German: *langsam*, *mittel*, *schnell*
LEVEL_ADAPT_SPEED_TXT/_IDX, -, evo: adapt_di.c (setLevelAdaptSpeed())

number of fitness classes **Notenklassenanzahl**

Adaptation of HyGLEAM can be performed on different fitness levels separately. These levels are called fitness classes. A parameter value of one means that there is only one class and the adaptation is done regardless of the fitness of the present individual. A more differentiated treatment is performed if 3, 4, or 5 classes are used. At present the default limits of of these classes can be altered by the interactive HyGLEAM version based on the textual interface only.
NUMB_OF_FCLASSES_TXT/_IDX, -, evo: adapt_di.c (set_std_nkl())

fraction of old LHC-distribution[%] **Anteil der alten LSV-Verteilung [%]**

Adaptation of the LHC distribution is based on the previous distribution and on the newly calculated one in such a way that the given fraction *frac in percent* of the old one is added to a percentage of (100 – *frac*) of the new one resulting in the final new distribution.
OLD_DISTR_FRAC_TXT/_IDX, oldDistribFrac, evo: evo_gsw.c

Application dependent parameters:

rotation angle in degrees **Drehwinkel in Grad**

Used to rotate one of the build-in mathematical test functions. Range: 0° - 180°
MBF_ROT_ANGLE_TXT/_IDX, -, simu: mbf_sim.c

with phenotypic repair **Mit phaenotypischer Reparatur**

Optional parameter, which has an effect only, if phenotypic repair is supported by the application at hand and its interpreter and/or simulator. The default value depends on this implementation and is set to FALSE, if there is no phenotypic repair implemented. The applicability of this parameter is controlled by "mitOptionPhaenoRep".
WITH_PHAENO_REP_TXT/_IDX, mitPhaenoRepair, appl: appl_if.c

genetic repair rate [%] **Genetic-Repair-Rate [%]**

Optional parameter, which has an effect only, if genotypic repair is supported by the application at hand and its interpreter and/or simulator. It controls the fraction of offspring, which are selected for repair. The default value depends on the present implementation and is set to zero, if there is no genotypic repair implemented. The applicability of this parameter is controlled by "mit_gen_rep_par".
GEN_REP_RATE_TXT/_IDX, gen_rep_par, appl: appl_if.c

simulator model **Simulatormodell**

String for the designation of an external simulator model. This can be a file name with path.
SIMU_MOD_NAME_TXT/_IDX, extSimuModelName, simu: simu.c

timeout for ext. simulator [sec] **Maximale Zeit pro ExtSimu-Lauf[sec]**

Time out for the communication with external simulators. There are different variables according to the external simulator interface used. The matlab interface does not use TMO.
SIMU_MAX_TIME_TXT/_IDX, variables set by `update_ext_simu_tmo()`

old pipe-based interface:	simu_first_waits, simu: ext_sim.c
ext. Simulationservices (ESS):	simuResultWaitCycles, simu: simu.c, extSimuServKoppl.c
Matlab interface:	simuResultWaitCycles simu: simu.c, matlab_koppl.c

runs per (re)init. of ext.simu. **Laeufe pro (Re)Init. des ExtSimu**

After the given number of simulations is reached the external simulator will be reinitialized. This feature was used for external simulators, which have a memory leak in the context of old pipe based interfaces.
SIMU_RUNS_PER_INIT_TXT/_IDX, extSimuRunLimit, simu: simu.c
run_ctr_limit, simu: mathkopl.c

with logging of ext.simu. interface **Mit Logging d.ExtSimu-Schnittstelle**

Optional parameter for logging the I/O of the interface to an external simulator. Usually used for debugging purposes.
WITH_SIMU_LOG_TXT/_IDX, mit_simulog, sys: file_gsw.c

with display of ext.simu runtime **Mit ExtSimu-Rechenzeit-Anzeige**

Optional parameter controlling the display of the run time of each external simulator run. Can be used for debugging purposes. Was used in the context of the old pipe based interface.
WITH_SIMU_TIME_TXT/_IDX, mit_rz, aufg: tsk_data.c

with Mitsubishi R500 **Mit Mitsubishi R500**

Parameter of the robot application LESAK/Plus [2, 3, 6], which activates the build-in simulation model of the Mitsubishi R500 industrial robot [5]. The selection of this robot model sets *number of axis* to 5.
LSKP_R500_TXT/_IDX, mitr500, simu/lskp: rob_gsw.c (define_mitsubishi()), rob_sim.c (update_rob_task())

with collision test **Mit Kollisionstest**

Parameter of the robot application LESAK/Plus, which controls the collision checking.
LSKP_COLLI_TEST_TXT/_IDX, kollitest, rob_gsw.c, rob_sim.c (update_rob_task())

with end stop test **Mit Anschlagtest**

Parameter of the robot application LESAK/Plus, which controls the checking for end stops of the robot axes.
LSKP_STOP_TEST_TXT/_IDX, anschlagtest, rob_gsw.c, rob_sim.c (update_rob_task())

with depreciation on stop/crash **Mit Abwertung bei Anschlag**

Parameter of the robot application LESAK/Plus, which controls the depreciation in case of collision or reaching an end stop.
LSKP_STOP_DEPREC_TXT/_IDX, anschlag_ko, rob_gsw.c, rob_sim.c (update_rob_task())

number of axes **Achsanzahl**

Parameter of the robot application LESAK/Plus, which sets the number of axes. Used for robots different from Mitsubishi R500.
LSKP_AXES_NUMBER_TXT/_IDX, axisanz, simu/lskp: rob_gsw.c (define_mitsubishi()), rob_sim.c (update_rob_task())

simulator cycle time [sec] **Simulatortakt in Sekunden**

Parameter of the robot application LESAK/Plus, which controls the cycle time of the internal robot simulator.
LSKP_CYCLE_LENGTH_TXT/_IDX, bildtakt, rob_gsw.c, rob_sim.c (update_rob_task())

CEC: limit of fct.evaluations (FEs) **CEC: Limit d. FktEvaluationen (FEs)**

This parameter is retained for compatibility with the CEC'05 benchmarks. It affects the same variable as *the limit for evaluations*.
CEC_FES_LIMIT_TXT/_IDX, defEvalLimit, evo: evo_gsw.c

Domain name of extern. SimuServices **Domainname der externen SimuDienste**

This parameter is required for the communication with the External Simulation Services (ESS).
SIMU_DOMAIN_NAME_TXT/_IDX, extSimServDomainName, simu: simu.c

Max. employee peak per shift **Soll-SchichtspitzenMax**

Optional parameter of the OPAL application (scheduling in process industry).
OPAL_MAX_SHIFTS_TXT/_IDX, sollSSpitzenMax, opal: opal_bew.c

Parameters of the command line version:

with detailed logging **Mit detaliertem Logging**

A more detailed logging of the given options, the optimization run, and its results. Used in the command line version only.
WITH_DETAILED_LOG_TXT/_IDX, detailed_log, evo: evo_anzg.c

with result simulation **Mit Ergebnis-Simulation**

The best result of the optimization run is simulated and the outcome is written to a text file with the name of the log file and the extension ".res". According to the application the run may produce additional result data for subsequence usage. This option requires that the *number of results* is at least 1. Used in the command line version only.
WITH_RES_CHR_SIMU_TXT/_IDX, withResChrSimu, main program

with saving of result chromosomes **Mit Rettung der ErgebnisChromosomen**

All result chromosomes specified by *number of results* are written to a chromosome file (portable text file variant). The file name is the name of the log file supplemented by "_res.aks". Used in the command line version only.
WITH_RES_CHR_SAVE_TXT/_IDX, withResChrSave, main program

with job termination tests **Mit Tests auf Jobabbruch**

The command line version may be stopped externally by the creation of file named "evo_stop.tmp" in its working directory. This feature can be activated by the parameter. If so, an appropriate test rate can be set by the "termination test rate [pairings]"-parameter.
WITH_TERM_TEST_TXT/_IDX, with_evo_term_tst, evo: evo_gsw.c

termination test rate [pairings] **Abbruch-Testrate [Paarungen]**

Depending on the setting of “with job termination tests”, the command line version may be stopped externally by the creation of file named "evo_stop.tmp" in its working directory. The parameter specifies the frequency in pairings with which the existence of the file is tested.
TERM_TEST_FREQ_TXT/_IDX, evo_tst_frequ, evo: evo_gsw.c

2.3 Error Messages and their Interpretation

GLEAM has three states for managing errors:

OK: In the normal state all menu functions are enabled and there are no restrictions in functionality.

ERROR: Some menu functions are locked. A possibly running optimization job or other functions are aborted. The error can be corrected by taking appropriate measures according to the situation.

FATAL: Locking of further menu functions. All running functions are aborted. The CLV terminates. The status can be reset using the menu function “IgnorFatal” of the “System” menu, see 4.2.9.12. This is intended for error analysis and should be used with caution because a program crash is possible depending on the error situation.

The messages start with the type of message: Message, ERROR, or FATAL, followed by a reference to the location of the error message call in brackets. After that the actual message is shown.

A message his something like a hint to the user. For example, if a gene model file contains parameterizations for LHCs and is loaded by GLEAM instead of HyGLEAM, these specifications are ignored, because GLEAM does not contain any local procedures or heuristics. The user is informed of this by the following message:

```
Message (HMOD_mod_data): Gene model expects not implemented local hill
                         climbers or heuristics.
```

The origin of this message is in module `mod_data.c` of package `hmod`.

A maximum of 30 (error) messages are stored and shown one after the other, starting with the first one. If more occur, which could happen e.g. during an optimization run, they are counted and ignored. The reason for this is that the first error messages reflect the actual cause and the subsequent ones usually do not contribute to a clarification. The following is an example for the occurrence of several messages. After the mathematical benchmark function *Griewank function* has been loaded as an application by a corresponding EXP file, an attempt is made to load a program parameter file for the robot application. Since this would set unsuitable parameters, only parameter files of the same application class (see Sect. 3.1 and 3.2 and [7]) are allowed. Therefore the following messages are displayed:

```
 *** 3 messages at state "ERROR":
ERROR (AUFG_tsk_data/201): TSK: Incompatible application: "LESAK/Plus"
ERROR (AUFG_tsk_data/202): Error in TSK file header!
Message (MEN_f_load): Old program parameters remain valid: "/home/wilfried/
                      gleam/InitFiles/mbf_griew_5par_clv_e.tsk"
```

The first one describes the error, in this case a different application class ID than the one actually loaded. The second one generalizes this to an invalid file header, while the last one tells the user that the already loaded program parameter file remains valid, thus maintaining consistency.

Error and fatal error messages have a more precise description of their origin by adding a distint number to the respective message call. The first two messages are from the module `tsk_data.c` of package `AUFG` and there from the calls with the unique numbers 201 and 202.

3 Initialization Files

There are the following mandatory initialization files, which are all text files:

1. The MOD file describing the gene model, see [7]
2. The BEW file describing the evaluation criteria, their normalization and possible penalty functions and their weights.
3. The TSK file setting several general program parameters, specific parameters for the simulators and parameters for the command line version (CLV). Many parameters can be altered either online by the interactive versions or by command line options of the CLVs.
4. The EVO file configuring mutation rates and the genetic operators [2, 3] and their probabilities of execution.
5. The error message file. At present, there are `ftext_gb.txt` and `ftext_d.txt` according to the language used. For the OPAL/V application there are the additional files `opftxt_e.txt` and `opftxt_gb.txt`.

The file specifications of the first four files are contained in the mandatory EXP file describing an optimization task referred to as experiment setting. The EXP file must be specified in the command line, when the program is called.

The fifth file must be accessible by the program. If not, the program terminates immediately. The file must be contained in the working directory or in a directory, the environment variable `GLEAM_ROOT` points at. It is recommended to create a directory for all initialization files, in which the error message file and all initialization files with which relevant optimization runs are performed are stored. When an experiment file is loaded, the program first checks whether the files it contains are in the working directory and if not, whether they are in the directory specified by `GLEAM_ROOT`, if this environment variable exists.

Example for an experiment file in English, assuming that `GLEAM_ROOT` exists or all files are in the working directory:

```
# ========================= Experiment Description ==========================
GLEAM/AE
# generated at Wed, 01.03.2017 12:11:40
# ---------------------------------------------------------------------------
Gene model           = mbf_griew_5par_lhc.mod
Evaluation           = mbf_griew_5par_e.bew
Program parameters   = mbf_griew_5par_clv_e.tsk
Evolution parameters = mbf_wpar_e.evo
Chromosome memory    =
```

and in German with absolute paths:

```
GLEAM/AE ===================================================================
# Experiment "mbf_griewank" vom 10.05.2007         File: mbf_griew_5par_clv.exp
#============================================================================
Genmodell           = /home/wilfried/gleam/InitFiles/mbf_griew_5par_lsv.mod
Bewertung           = /home/wilfried/gleam/InitFiles/mbf_griew_5par.bew
Programmparameter   = /home/wilfried/gleam/InitFiles/mbf_griew_5par_clv.tsk
Evolutionsparameter = /home/wilfried/gleam/InitFiles/mbf_wpar.evo
```

Strings starting with a "#"-sign are treated as comments. The first entry of an EXP file must be the program ID "`GLEAM/AE`". The remainder of the line containing the ID is skipped. Standard entries are assignments of a value to a name, here strings of file specifications to the corresponding program parameters. Note that their names may contain blanks. Usually, the next four entries specify the initialization files listed above. If such an assignment is repeated the last value applies.

The English example contains a further assignment in the last line, which is here empty. If a chromosome file (MEM file, [9]) is specified, it will be read after the standard initialization files. MEM files are expected to be binary files, unless specified in the TSK file otherwise. As the assignment is empty in the given example no MEM file will be read.

EXP files may contain additional initialization files required by the evaluation part of a specific application as specified in the MOD file. These files are listed in a special section after the standard files, which is started by a special comment beginning with the string "`#!`". An example is given below.

```
# ========================== Experiment Description ==========================
GLEAM/AE
# written at Wed, 01.03.2017 12:11:40
# ------------------------------------------------------------------------------
Gene model                = mitsu_hy_e.mod
Evaluation                = lesak_e.bew
Program parameters        = mitsu_e.tsk
Evolution parameters      = lsk_stnd_e.evo
Chromosome memory         = myLastChromsomes.mem
#! ----------------- optional list of application specific files --------------
mitsu.kin       #  kinematics of the robot
stolper.obs     #  obstacles
```

If the environment variable `GLEAM_ROOT` is defined and contains e.g. the directory

`/home/wilfried/gleam/InitFiles`,

all files specified in the EXP file are first searched in the working directory and then in `/home/wilfried/gleam/InitFiles`. In the log file they are entered as they are found. This also applies to the display functions of the interactive program versions. In the above example, all files could be located in the directory designated by `GLEAM_ROOT`, while myLastChromsomes.mem, for example, is from the last run and is located in the working directory.

3.1 Gene Models (MOD Files)

The gene model file contains the *application class ID*, which must be the same as in the corresponding TSK file. It also contains the number of application specific additional files, which must be specified in the EXP file and some information about them. There is a separate MOD file documentation, see [7].

3.2 Program Parameters (TSK Files)

For TSK files the same general rules regarding comments and assignments apply as for EXP files. The list of possible program parameters is described in detail in Sect. 2.2.

For the interactive versions most the parameters may be altered by the general `ProgParam` menu or by dialogs or sub-menus of the `Evo/Opt` menu. As far as dialogs are concerned the program parameters serve as default values.

In the command line version (CLV) many program parameters can be overwritten by the corresponding command line options, see Sect. 2.1.

A simple TSK file, which is used in conjunction with the build in mathematical test functions as specified by the application class ID `MathFkt`, is given below. It just rotates the function by 30° to make it harder.

```
# --------------------- Program Parameter File (TSK file) ---------------------
GLEAM/AE  MathFkt
# written on Wed, 16.11.2016
# ------------------------------------------------------------------------------
rotation angle in degrees     = 30.0
# ------------------------------------------------------------------------------
```

Note that the first two entries must be

- the program ID `GLEAM/AE`
- the application class ID, which must be the same as specified in the related MOD file.

A more complex TSK file, which is used to initialize a CLV run for a mathematical test function is as follows:

```
# ------------------------ Program Parameter File (TSK file) ------------------------
GLEAM/AE  MathFkt
# written on 31.7.2020
# used for the Griewank test function with 5 parameters
# ------------------------------------------------------------------------------------
with init. of random generator    = yes
with detailed logging             = yes    # only for command line version
with result simulation            = 1      # only for command line version
with saving of result chromosomes = No     # only for command line version
with job termination tests        = yes
termination test rate [pairings]  = 20
# ------------------------------------------------------------
optimization procedure             = GLEAM
population size                    =    300
deme size                          =      8    # (Pop|Deme): (5|4) (10|6) (>=20|8)
init. strategy for start population= new
init. strategy parameter           =      0
minimal hamming dist.f.XO/Reco [%] =      0.1
ranking parameter                  =      1.5  # 1.0 - 2.0 (selective pressure: min - max)
acceptance rule for offspring      = localLeast-ES
number of digits of (simu)parameter=      7
# ------------------------------------------------------------
run time limit in seconds           =    120    # 2 min
limit for generations               =  10000
max.gen. w/o deme improvement (GDI) =    999    # GDI
max.gen. w/o child accept. in deme  =    300    # GAc
target fitness                      = 100000.0
```

For historical reasons, a TSK file may also contain application-specific data at the end. These data are separated from the standard part by a special comment starting with "#!" like in the EXP file. This is only used by the robot application and all further is described in Sect. 3 in [5].

3.3 Fitness Calculation (BEW-Files)

Although BEW files are text files, you should not edit them, but create an evaluation using an interactive GLEAM version. For this purpose the experiment for the Griewank test function `mbf_griew_5par_clv.exp`, which is contained in the `testfield` sub directories of the `/<local_root>/gleam/linux/gleam_ae` and `/<local_root>/gleam/linux/hy_gleam_a` directories, can be loaded. The files to be loaded must be contained in the directory `/<local_root>/gleam/InitFiles` or the EXP file must be edited accordingly. This experiment will be used as an example for the description of the user functions in Sect. 4, where the construction of a new assessment is described in Sect. 4.2.6 in detail. As with all test functions included in GLEAM, the Griewank function uses only one criterion and can therefore very well serve as a starting point for a new extended evaluation for a different application.

Fitness calculation in GLEAM is based on the cascaded weighted sum, see [2, 10].

3.4 Production of Offspring and Evolution Parameters (EVO-Files)

The EVO files are used to configure the application of genetic operators for the offspring generation of GLEAM. This paper explains how the genetic operators in GLEAM are used to produce offspring. The details of the associated configuration by an EVO file is covered in [11].

The details of the generation of offspring are related in content to the chromosome type selected in the MOD file, see also [3, 7] and [2, sections 4.2.2, 4.3 and 4.4]. This correlation, i.e. the suitability of the genetic operators used for the current chromosome type, is not checked by GLEAM and it is left to the user to observe the dependencies described in [2, Sect. 4.4] and [3].

In GLEAM, over the course of a generation, each individual becomes a parent, chooses a partner and produces one or usually several offspring. For this purpose groups of genetic operators are formed, which must consist of at least one operator. Each operator is given a probability of execution and they are executed in the order given in the group list. Each group also has a group probability and the groups are executed in sequence according to their group probability.

One or two offspring can be produced per group. This is determined when a group is parameterized, assuming that this determination matches the genetic operators of the group. The following should be observed:

1. If **two descendants** are to be created, the first operator in the group list must be a crossover or recombination operator[1]. Any subsequent operators must be mutation operators and they are applied to one of the offspring in the order of the list and according to their probability. The other offspring of the crossover or recombination is not mutated and remains unchanged.

2. If a single descendant is to be created, a clone of the parent (copy) is created and modified using the operators in the list. In this case, the list may only contain mutation operators. It may happen that the clone remains unchanged due to probabilities. In this case, it is deleted and thus one descendant less is created.

Warning: These two restrictions must be observed. It has never been tested what happens if, for example, you want to create two offspring with mutations only or if you place the crossover or recombination operator not at the top of the list or if you write more than one crossover or recombination operator to the list!

There is a maximum of 10 operator groups (constant `ANZ_PAAR_OPS` in `evo.h`). Each operator group can consist of a maximum of 16 genetic operators (constant `ANZ_GEN_OPS` in `go_gsw.c`).

Since it does not make sense to create offspring by crossover or recombination from identical or very similar parents, the execution of these operators can be controlled by the Hamming distance of the parents. The control is done via a corresponding parameter of an optimization job (see Sect. 4.2.8), the default value of which can be set as program parameter, see Sect. 2.2, parameter "`minimal hamming dist.f.XO/Reco [%]`". If there is no crossover due to too great a similarity of the parents, no descendants are generated and any subsequent mutations of the affected operator groups are therefore also omitted. The Hamming distance of two non-empty chromosomes is defined for the three different chromosome types of GLEAM [2, 3] by different measures as follows.

The parameter distance $\Delta_{par}(AK_1, AK_2)$ of two chromosomes AK_1 and AK_2, which are both not empty, is defined for all parameters, for which the upper limit *og* is greater than the lower one, as:

$$\Delta_{par}(AK_1, AK_2) = \frac{1}{anz} \sum_{1}^{anz} \frac{|param_{i,1} - param_{i,2}|}{og_i - ug_i}$$

where $param_{i,j}$: value of the *i*-th parameter in chromosome AK_j. The parameters are numbered in the sequence of the gene type definitions.

ug_i, og_i: lower and upper bound of the range of values of the *i*-th parameter with $ug < og$

anz: number of all parameters of all genes

This is the parameter distance for chromosomes of type 1 or 2, i.e. chromosomes with fixed length.

1: Recombination in GLEAM refers to all forms of n-point crossover, see also [2, 3].

For chromosome type 2, where the gene sequence is relevant, an additional measure of the positional differences of the genes $\Delta_{pos}(AK_1, AK_2)$ is required:

$$\Delta_{pos}(AK_1, AK_2) = \frac{1}{abst_{max}} \sum_{1}^{len} |I_1(A_i) - I_2(A_i)|$$

$$abst_{max} = \frac{len^2}{2} \qquad len \text{ even}$$

$$abst_{max} = \frac{len^2 - 1}{2} \qquad len \text{ odd}$$

where *len*: length of a chromosome with $len > 1$

$abst_{max}$: distance maximum of all genes within one chromosome

The total distance is the mean value of $\Delta_{par}(AK_1, AK_2)$ and $\Delta_{pos}(AK_1, AK_2)$.

For chromosomes of variable length and relevant gene sequence (type 3) the difference of the presence of genes $\Delta_{pres}(AK_1, AK_2)$ must be additionally measured. The goal is to determine the precise difference of similar chromosomes while the exact value of discrepancy of dissimilar ones is of less interest. So the resulting measure can be imprecise for more different chromosomes, thus resulting in a less complex formula which reduces the computational effort for the fairly frequent distance calculations.

Let AK_1 be the longer chromosome and G_{com} the set of genes common to two non empty chromosomes AK_1 and AK_2. As genes in chromosomes of variable length can occur several times, they are treated in the sequence of their indexing. If G_{com} is empty, the overall distance is set to the maximum value of 1. Otherwise, Δ_{par} and Δ_{pos} are defined over the set of common genes G_{com}. $abst_{max}$ is taken from AK_2. This may lead to a to great value of Δ_{pos}, which may increase above 1 especially for chromosomes with large differences. Thus Δ_{pos} is limited to 1 and the error is accepted as it increases with the chromosome difference.

$$\Delta_{pres}(AK_1, AK_2) = 1 - \frac{card(G_{com})}{max(len(AK_1), len(AK_2))}$$

where $len(AK_i)$: length of chromosome AK_i

card(A): number of elements of a set *A*

The distance measures Δ_{par} and Δ_{pos} are calculated for $G_{com.}$ As differences of the presence of genes are of greater importance than the other two measures, the overall distance is again the mean value of the three distances, but with Δ_{pres} weighted three times.

The proofs of the metric properties of the three distance measures can be found in [12].

4 Version with a Textual User Interface

The version with a textual user interface is called with a maximum of two program parameters:

`hy_gleam_a [<exp-file> [<log-file>]]` for HyGLEAM and for GLEAM:

`gleam_ae [<exp-file> [<log-file>]]`

The specification of both files can be done with or without extension (`.exp`, `.log`). If no **`<log-file>`** is specified, the name of the experiment file will be used for the log file instead, provided the experiment file exists. Otherwise, **`gleam_ae.log`** is used. If no **`<exp-file>`** is specified, **`default.exp`** and **`gleam_ae.log`** will be used. The program will be terminated, if the experiment file or a file specified therein does not exist.

After program start an introductionary text frame is displayed, which is followed by the initialization report and the main menu:

```
######################################################################
##                                                                  ##
##                       G L E A M  /  A E                          ##
##                       =================                          ##
##                                                                  ##
##      General Learning Evolutionary Algorithm and Method          ##
##                    Application Environment                       ##
##                                                                  ##
##  GLEAM is an EA of its own, which combines elements of comp.     ##
##  sci., of the ES of Rechenberg and of the GAs of Holland.        ##
##  Applications implemented in this version: Collision-free        ##
##  path planning for industrial robots and several mathema-        ##
##  tical benchmark functions.                                      ##
##  This program is Free Software published under the GNU LGPL,     ##
##  see menu item Info/Version or sources of the main programs.     ##
##                                                                  ##
##      Prof. Dr. Christian Blume (HS Köln, Campus Gummersbach)     ##
##      Dr. Wilfried Jakob (KIT, Campus North, IAI)                 ##
##                                                                  ##
##                          GLEAM/AE V2.2.3/Off from 09/16/2020     ##
######################################################################

Initialization:
===============
  The initialization files are also searched in the GLEAM_ROOT directory: "/home/wilfried/gleam/InitFiles"
  Gene Model "/home/wilfried/gleam/InitFiles/mbf_fox_e.mod" loaded.
  Evaluation "/home/wilfried/gleam/InitFiles/mbf_fox_e.bew" loaded.
  ProgParams "/home/wilfried/gleam/InitFiles/mbf_fox_rot_e.tsk" loaded.
  EvoParams  "/home/wilfried/gleam/InitFiles/mbf_wpar_e.evo" loaded.
  No chromosome file read.
  1 messages at state "OK": Message (HMOD_DATA): Gene model expects not implemented
                            local hill climbers or heuristics
Initialization completed.

------------------------- Main Menue -------------------------
1: Evo/Opt      2: Simulator    3: Info
5: ProgParam    6: Evaluation   7: Load         8: Save
9: System                                     C: Quit
--------------------------------------------------------------
Select menu item:
```

The line indicating the existence of the environment variable `GLEAM_ROOT` and its contents is only displayed if `GLEAM_ROOT` exists. After completing the initialization possible (error) messages are displayed. In this case its only a message giving the hint, that there are parameterizations for local hill climbers in the gene model file, which are ignored by pure GLEAM. There are three error levels in GLEAM: *OK*, *error* and *fatal error*. If the latter does not cause a program termination some interactive functions are disabled.

4.1 Program Parameters

Many TSK file parameters are intended for the CLVs. To ensure that the TSK files can be exchanged between the CLVs and interactive versions, the TSK file parameters are loaded as default values for the interactive programs. They can be changed either by the `ProgParam` menu OR by the `Opt Params` menu of the `Evo/Opt` menu and to a lesser extent by the `Simulator` menu. Changes to TSK file parameters in the listed menus can be saved to a TSK file. Only parameters that have been read in or changed before are saved. An exception to this is the specification of the optimization method, which is usually saved regardless of whether this parameter has been read in or changed.

As a general rule, TSK file specifications overwrite MOD file specifications (e.g. LSV termination values of the LSV selected in the TSK file). It was not investigated whether and to what extent inconsistencies can arise due to unsuitable multiple loading of TSK and MOD files.

The editability mentioned in table 1 refers to the interactive program versions as follows: “yes” means that it is editable by the “ProgParam”-menu, “optP” indicates editability by the “Opt Params”-menu of the “Evo/Opt”-menu, while “simu” means changeability by the “Simulator”-menu. Brackets indicate that the availability of this parameter depend on the actual application or GLEAM version. E.g., some parameters are meaningful for the HyGLEAM extension only.

4.2 Menus

First some general notes on the menus and dialogs:

- Every menu and dialog can be aborted by the escape key. In case of a function dialog this function will not be executed.
- Input like strings or numbers can be edited and the following keys are recognized: left and right arrow, begin and end of line, and delete and backspace. The input is terminated as usually by the enter key.

User input is marked by a green background. An empty input is indicated by the return sign: ↵

Many menu functions contain dialogs. In these dialogs, a possible value range is given in round brackets "(...)" and the default or old value in square brackets "[...]". The latter can be accepted by pressing the return key.

4.2.1 Main Menu

The main menu shown at the beginning of the chapter contains the following menu items:

- Evo/Opt: Definition, parameterization, management and execution of optimization runs
- Simulator: Evaluation of a chromosome, which is usually done by simulation.
- Info: Information about the chromosomes stored in the system, the results of previous runs, individual chromosomes, the loaded files and the current program application
- ProgParam: Display and modification of program and strategy parameters of optimization procedures
- Evaluation: Display and change of the evaluation based on the cascaded weighted sum
- Load: Loading files, including a new experiment
- Save: Saving to files. This includes a changed evaluation or of chromosomes.

- System: Display of memory usage, deletion of single chromosomes or the complete chromosome memory, chromosome editor, various test functions

4.2.2 Program Parameter Menu

The size of the program parameter list depends on the current program application. The number program parameters is displayed when the menu item is called up followed by the sub menu:

```
There are a total of 54 defined program parameters.

------------------------ ProgParam -------------------------
1: ShowParam    2: ChngParam    3: All Param    4: Close
------------------------------------------------------------
Select menu item: 1
```

The first menu item outputs a list of editable parameters, here the one of `gleam_ae` with an experiment for test functions. Therefore, the parameter "`Rotation angle in degrees`" is at the end of the list. It is used to make the test functions more difficult.

```
     Program-Parameter                 Value
with init. of random generator      =    yes
with recording of best of evolution =    yes
with recording of statistical data  =    yes
with stat.data recording per gen.   =     no
population size                     =     90
init. strategy parameter            =      0
number of results                   =      1
run time limit in seconds           =    300
limit for generations               =   2000
max.gen. w/o child accept. in deme  =    100
max.gen. w/o deme improvement (GDI) =    400
limit for evaluations               = 2147483647
backup rate [generations]           = 2147483647
number of digits of (simu)parameter =      7
init. strategy for start population =    new
target fitness                      = 100000
rotation angle in degrees           = 30
```

The two `INT_MAX` values de facto disable the termination of an optimization run by exceeding an evaluation limit and creating backups of the current population every n generations.

The second menu item allows to change a parameter, here e.g. the generation limit. To allow a selection, numbers are assigned to the parameters:

```
 Modification of current Program Parameters:

No. Program-Parameter                   Value
 1: with init. of random generator      =    yes
 2: with recording of best of evolution =    yes
 3: with recording of statistical data  =    yes
 4: with stat.data recording per gen.   =     no
 5: population size                     =     90
 6: init. strategy parameter            =      0
 7: number of results                   =      1
 8: run time limit in seconds           =    300
 9: limit for generations               =   2000
10: max.gen. w/o child accept. in deme  =    100
11: max.gen. w/o deme improvement (GDI) =    400
12: limit for evaluations               = 2147483647
13: backup rate [generations]           = 2147483647
```

```
14: number of digits of (simu)parameter =      7
15: init. strategy for start population =    new
16: target fitness                      = 100000
17: rotation angle in degrees           = 30
parameter number (1..17): 9
limit for generations (0 .. 2147483647) [2000]: 500
```

4.2.3 Save Menu

The menus Save and Load have the same structure, i.e. the same file categories are in the same place. Since the file type is fixed after selecting a menu item, only the file name with optional path is regularly requested. The following can be saved:

- the complete experiment with all saveable files belonging to the experiment as far as the corresponding changes have been made. At least one EXP file (.exp) is created and optionally further files for evaluation, program parameters and chromosome memory.
- the evaluation in a BEW-file
- the program parameters in a TSK file
- the entire chromosome memory (AK-Mem) in one MEM file
- selected chromosomes in an AKS file

The differences and similarities between MEM and AKS files are discussed in detail in the introduction of [9] and knowledge of the contents of this section is highly recommended.

A MEM-file and an AKS-file will be created as representative for the other files and finally the whole experiment will be saved.

```
---------------------------- Save -----------------------------
1: Experiment                    3: Evaluation   4: ProgParams
                6: Chr-Memory    7: Chromosoms   8: Close
---------------------------------------------------------------
Select menu item: 6

                     Save Chromosome Memory:

"mem"-FileSpec (without ext): temp
Standard chromosome file (binary) [Y/n]: ↵

Save chromosomes to file "temp.mem":
    2 chromosomes of fitness class  8 saved.
    1 chromosomes of fitness class 10 saved.
    1 chromosomes of fitness class 12 saved.
    1 chromosomes of fitness class 13 saved.
Total of 5  chromosomes successfully saved.
```

After the input of the file name (here "temp") you are asked whether a binary or textual MEM-file should be created. See the explanations in [9]. In the example a binary file is created. After saving, a list of the saved chromosomes per fitness class and the total number of saved chromosomes will be displayed.

The chromosome memory is divided into 16 fitness classes. The address of a chromosome consists of its class and a consecutive number under which it is stored in that class, see also the table in the introduction. Thus, from the above list of the saving example it can be seen that there are e.g. two chromosomes with a fitness assigned to fitness class 8, that are saved. The corresponding addresses then are 8/1 and 8/2. As the next higher class 10 contains one chromosome, it has the addresses 10/1. With respect to the fitness ranges assigned to the classes, please refer to the following backup example and to Sect. 4.2.5. Fitness class 1 is intended for unrated or extremely poorly rated chromosomes.

If only single chromosomes shall be saved, select the menu item "Chromosoms":

```
--------------------------- Save ---------------------------
1: Experiment                    3: Evaluation   4: ProgParams
               6: Chr-Memory   7: Chromosoms   8: Close
------------------------------------------------------------
Select menu item: 7

                        Save Chromosomes:

"aks"-FileSpec (without ext): temp

  Class    Fitness Interval  Number
     1          0 -      0      0
     2          1 -   2499      0
     3       2500 -   4999      0
     4       5000 -   9999      0
     5      10000 -  19999      0
     6      20000 -  29999      0
     7      30000 -  39999      0
     8      40000 -  49999      2
     9      50000 -  59999      1
    10      60000 -  69999      0
    11      70000 -  74999      0
    12      75000 -  79999      1
    13      80000 -  84999      1
    14      85000 -  89999      0
    15      90000 -  94999      0
    16      95000 - 100000      0

                        Save Chromosomes:
Chromosomes to be saved can be marked via the following dialogs.
Class = 0: End of dialog and saving the marked chromosomes.
Index = 0: Mark all chromosomes of the typed in fitness class.
ESC      : Abandon function, reset marks and save nothing.
Class and index of the chromosome to be saved: 8
Class and index of the chromosome to be saved: 8/ 2
Chromosome 8/2 has been marked for saving.
Class and index of the chromosome to be saved: 13
Class and index of the chromosome to be saved: 13/ 1
Chromosome 13/1 has been marked for saving.
Class and index of the chromosome to be saved: 0

Standard chromosome file (binary) [Y/n]: ↵

                        Save Chromosomes:
Save chromosomes to file "temp.aks":
    1 chromosomes of fitness class  8 saved.
    1 chromosomes of fitness class 13 saved.
Total of 2 chromosomes successfully saved.
```

First the file name is entered (here "temp").

This is followed by a list of fitness classes and the number of chromosomes stored in them, as well as instructions on how to mark the chromosomes for saving.

Then, the chromosomes 8/2 and 13/1 are selected, and the selection dialog is terminated by entering a 0 for the class.

Again, the file type is selected.

The result of the saving is then documented by corresponding outputs.

When saving the experiment, the currently loaded files are assumed and any changes to the respective data sets are taken into account in the dialogs.

```
--------------------------- Save ---------------------------
1: Experiment                    3: Evaluation   4: ProgParams
               6: Chr-Memory   7: Chromosoms   8: Close
------------------------------------------------------------
Select menu item: 1

                         Save Experiment:

"exp"-FileSpec (without ext): demo
Does evaluation conform to the one of file "/home/wilfried/gleam/InitFiles/
    mbf_fox_e.bew"? [Y/n]: ↵
Program parameters possibly changed.
Do the program parameters conform to those of file "/home/wilfried/
                                                    gleam/InitFiles/
    mbf_fox_rot_e.tsk"? [y/N]: ↵
"tsk"-FileSpec (without ext) [demo]: ↵
Save Chromosome memory? [Y/n]: ↵

                         Save Experiment:
Experiment saved in file "demo.exp":
  ProgParams saved in file "demo.tsk".
  Save chromosomes to file "demo.mem":
      2 chromosomes of fitness class  8 saved.
      1 chromosomes of fitness class  9 saved.
      1 chromosomes of fitness class 12 saved.
      1 chromosomes of fitness class 13 saved.
  Total of 5 chromosomes successfully saved.
File successfully saved
```

After entering the name of the experiment file (here "demo"), the queries are made according to the menus visited and entries made in the meantime: Since obviously nothing was changed in the evaluation, the file specification of the currently loaded evaluation file is offered.

Afterwards, a message is displayed if the program parameters have been changed. Accordingly, the default for the following query for the actuality of the loaded TSK file is set to "N" for No. The No is confirmed by the user in the example and a new TSK file is created with the name of the experiment. Of course, the user could have entered a different file name.

Finally, the user requests the saving of the chromosome memory.

This completes all entries and the saving can be carried out and the result displayed. The two written files are as follows:

```
# ======================  Experiment File List  ========================
GLEAM/AE
# written on Sat, 12.09.2020 15:59:10
# ---------------------- List of Standard Files: ----------------------
Gene model                  = /home/wilfried/gleam/InitFiles/mbf_fox_e.mod
Evaluation                  = /home/wilfried/gleam/InitFiles/mbf_fox_e.bew
Program parameters          = demo.tsk
Evolution parameters        = /home/wilfried/gleam/InitFiles/mbf_wpar_e.evo
Chromosome memory           = demo.mem
```

and

```
# ----------------- Program Parameter File (TSK file) -------------------
GLEAM/AE  MathFkt
# written on Sat, 12.09.2020 15:59:10
-------------------------------------------------------------------------
```

```
optimization procedure              =  GLEAM   # (0)
population size                     =     90
run time limit in seconds           =    300
limit for generations               =    500
rotation angle in degrees           = 30.0
```

4.2.4 Load Menu

Regarding addressing of chromosomes, please refer to the previous section. As the experiment loading generates most of the output when using the "Load" menu, it shall serve as a substitute for the other loading functions.

```
---------------------------- Load -----------------------------
1: Experiment   2: Gene Model   3: Evaluation   4: ProgParams
5: EvoParams    6: Chr-Memory   7: Chromosoms   8: Close
---------------------------------------------------------------
Select menu item: 1

                          Load Experiment:

Directory (CR=current): ↵
"exp"-Files in "/home/wilfried/gleam/linux/gleam_ae/testfield":
   1: mbf_fox_rot.exp
   2: lsk-test.exp
   3: demo.exp
   4: lesak.exp
   5: lesak_e.exp
   6: mbf_griew_5par_clv.exp
   7: mbf_griew_5par_clv_e.exp
   8: mbf_fox_rot_e.exp
File selection (1..8): 3

                          Load Experiment:
Experiment "demo.exp":
  Gene Model "/home/wilfried/gleam/InitFiles/mbf_fox_e.mod" loaded.
  Evaluation "/home/wilfried/gleam/InitFiles/mbf_fox_e.bew" loaded.
  ProgParams "demo.tsk" loaded.
  EvoParams  "/home/wilfried/gleam/InitFiles/mbf_wpar_e.evo" loaded.
Display loaded chromosomes? [y/N]: y
Chromosomes from file "demo.mem":
  AK stored at  8/1
  AK stored at  8/2
  AK stored at  9/1
  AK stored at 12/1
  AK stored at 13/1
  5 chromosomes successfully restored! Best fitness: 80611.4
Experiment successfully loaded.
```

After selecting the directory, a list of all files in question is generated, preceded by a consecutive number. The file selection is then made using this number.

Then the information of the loaded files follows similar to the program start.

Afterwards you can choose whether the addresses of the loaded chromosomes shall be listed or not. As this was desired in the example, a list of the loaded chromosomes and the best fitness is displayed.

4.2.5 Info Menu

The Info menu contains a number of information functions, especially concerning the chromosome memory (Chr-Mem to ChrLengths), individual chromosomes (ChrDisplay), the results of previous runs (OptResults), the loaded files (Load state) and the current program version (Version). Below are some examples of the dialogs and outputs.

The menu item "Chr-Mem" gives an overview of the chromosome memory sorted by fitness classes (see also Sect. 4.2.4):

```
---------------------------- Info -----------------------------
1: Chr-Mem      2: Chr/C Info   3: ChrLengths   4: ChrDisplay
5: OptResults   6: Load State   7: Version      8: Close
---------------------------------------------------------------
Select menu item: 1

Number of Chromosomes per Fitness Class

  Class    Fitness Interval  Number
     1          0 -      0      0
     2          1 -   2499      0
     3       2500 -   4999      0
     4       5000 -   9999      0
     5      10000 -  19999      0
     6      20000 -  29999      0
     7      30000 -  39999      0
     8      40000 -  49999      2
     9      50000 -  59999      1
    10      60000 -  69999      0
    11      70000 -  74999      0
    12      75000 -  79999      1
    13      80000 -  84999      1
    14      85000 -  89999      0
    15      90000 -  94999      0
    16      95000 - 100000      0

5 chromosomes.
```

The menu item "Chr/C Info" gives more detailed information about the chromosomes of a class:

```
---------------------------- Info -----------------------------
1: Chr-Mem      2: Chr/C Info   3: ChrLengths   4: ChrDisplay
5: OptResults   6: Load State   7: Version      8: Close
---------------------------------------------------------------
Select menu item: 2

Propertiess of the Chromosomes of a Fitness Class

Class (1..16): 8

Properties of the Chromosomes of Class 8:
  Addr    Fitness    Len    Segm    Sim    Ref
  8/1     45552.4      2       2     +       1
  8/2     40321.2      2       2     +       1
```

In addition to the fitness, the chromosome length (Len) and the number of segments (Segm) [2, 3] are shown. The column "Sim" indicates whether the chromosome has been simulated. The column "Ref" indicates the number of references to a chromosome and is only of importance for test purposes.

The data of a chromosome are listed with "ChrDisplay". The output can be done on the screen or to a file. In this example, a chromosome generated for the robot application [2, 3, 6, 5] is shown. After selecting the chromosome (here (10/1) the output destination can be specified and the start element of the gene list.

```
---------------------------- Info -----------------------------
1: Chr-Mem      2: Chr/C Info   3: ChrLengths   4: ChrDisplay
5: OptResults   6: Load State   7: Version      8: Close
---------------------------------------------------------------
Select menu item: 4

                       Chromosome Display:

Chr [class/index]: 10
Chr [class/index]: 10/ 1

                          Chain Display:

Output to s=screen, f=file, ESC=abort.  Select [s]: ↵
Enter start index (0=Hdr..13) [0]: ↵

Chromosome 10/1 of length 13:

Header data:
  fitness=62239.4284  simu=yes  refs=1  state=4  segm=5

  1.: SetMotorSpeed_M1:
ramp = 1.375177e+01                          voltVal_1 = 9.454131e+01 V

  2.: SetMotorSpeed_M3:
ramp = 1.359952e+01                          voltVal_3 = 1.210081e+01 V

  3.: Unchanged:
cycles = 2

  4.: SetMotorSpeed_M5:
ramp = 9.377992e+00                          voltVal_5 = -9.394423e+00 V

  5.: SetMotorSpeed_M2:
ramp = 1.452772e+01                          voltVal_2 = -2.989792e+01 V

  6.: SetMotorSpeed_M2:
ramp = 8.486883e-01                          voltVal_2 = 1.931222e+01 V

  7.: SetMotorSpeed_M1:
ramp = 2.545155e-01                          voltVal_1 = 9.904132e+01 V

  8.: SetMotorSpeed_M5
ramp = 4.041186e+00                          voltVal_5 = 7.668863e+01 V

  9.: SetMotorSpeed_M3:
ramp = 1.263162e+01                          voltVal_3 = 1.079874e+02 V

 10.: SetMotorSpeed_M2:
ramp = 9.454207e+00                          voltVal_2 = -7.526005e+00 V

 11.: Motor1_off:
ramp = 9.248674e+00

 12.: SetMotorSpeed_M2:
ramp = 2.213106e+00                          voltVal_2 = 1.010847e+02 V

 13.: Unchanged:
cycles = 5
```

The menu item "OptResults" displays the results of all optimization runs performed so far. In the example these were 4 very short runs for the robot task [2, 3, 6].

```
---------------------------- Info -----------------------------
1: Chr-Mem      2: Chr/C Info   3: ChrLengths   4: ChrDisplay
5: OptResults   6: Load State   7: Version      8: Close
---------------------------------------------------------------
Select menu item: 5

    Results of the Jobs Processed:

      4 Processed OptJobs with:
Populations          :      4
Generations          :     37
Evaluated offsprings:  24929
Fitness of the best :  62239.4  FitnClass: 10
Result chromosomes  :      4
Total time           : 00:00:04
---------------------------------------
Show optimization results? [Y/n]: ↵

                Results of the Jobs Processed:
                   (Fitness:Class/Index)
 55940: 9/1   57877: 9/2   58150: 9/3   62239:10/1
Reset optimization results? [y/N]: ↵
```

„Load State" gives an overview of the loaded files, the simulator used and the currently set topology of the neighborhood model. The current implementation supports only the ring structure with symmetrical neighborhoods.

```
---------------------------- Info -----------------------------
1: Chr-Mem      2: Chr/C Info   3: ChrLengths   4: ChrDisplay
5: OptResults   6: Load State   7: Version      8: Close
---------------------------------------------------------------
Select menu item: 6

       Loaded initialization files and status information:

Present applic. : Mathematical Benchmark Function
Experiment      : "demo.exp"
Gene model      : "/home/wilfried/gleam/InitFiles/mbf_fox_e.mod"
Simulator       : internal benchmark function: "MBF-Foxholes"
Evaluation      : "/home/wilfried/gleam/InitFiles/mbf_fox_e.bew"
Program param.  : "demo.tsk"
Evo param file  : "/home/wilfried/gleam/InitFiles/mbf_wpar_e.evo"
Chromosome file : "demo.mem"
Topology (par)  : symmetrical ring
```

4.2.6 Evaluation Menu

GLEAM uses the weighted sum extension described in [10] as *cascaded weighted sum*. If only the pure weighted sum is to be used, all criteria and possible penalty functions must be assigned to the highest priority level 1. The fulfillment values corresponding to the ε_i in [10] are then irrelevant. For formal reasons they must be set however in each case to a value in the evaluated interval.

The fitness values in GLEAM are between 0 and 100,000, with 100,000 being the best fitness. The weights of the criteria are given as fitness shares. For example, a fitness share of 60,000 corresponds to a weight of 60%.

The number of criteria can be less than the number of values returned by the simulation, which is given by the <akt_roh_erg_werte> entry in the MOD file, see also [7]. But it must not be bigger in any case! The order of the criteria managed by the "Evaluation" menu must match the order of the result values of a simulation.[1]

The functionality of the menu is explained by the extension of the evaluation of a mathematical test function (here the function according to Griewank) by a further criterion with a penalty function. It should be noted that this extension makes no sense in the context of a test function and serves only to describe the extension of an evaluation by a new criterion. First of all, after selecting the "Evaluation" entry in the main menu and the “Show Crit.” item, you get an overview of the currently defined evaluation:

```
------------------------ Evaluation ------------------------
1: Show Crit.    2: ChangeCrit    3: New Crit.
5: ShowEvalF     6: ChangeEval    7: PlotEvalF
9: ShowPenFct    A: ChangePenF    B: PlotPenFct
D: Plot->File    E: Quit Plots    F: Close
------------------------------------------------------------
Select menu item: 1

                          Assessment Criteria:

Nr. Criterion             Prio.     Fitness    Fulfillment      Penalty Fmax
                          Class       Share   W.Fitn    Value PrioClass Cont
 1: Griewank fct value :    1        100000    99999   0.0001   inactive  yes
-------------------------------------------------------------------------
                                     100000
                                     ======
```

As usual with mathematical test functions, only one criterion is defined without a penalty function. On the far left of the table is the number of the criterion, which is often referred to in the dialogs. This numbering also corresponds to the sequence of the evaluation criteria as it is to be returned by the evaluation component, e.g. a simulator.

The entry "Fmax Cont" on the far right of the table refers to an extension that has so far only been used in one application and is therefore not described further. There should always be a "yes" here.

Now a further criterion "temperature" with a weight of 40% and a penalty function if a maximum value t_{max} is exceeded shall be added, see Figure 4.1. In the example it is assumed, that a temperature in the range of t_{min} to t_{max} shall be reached as low as possible. Values up to t_{ok} are unproblematic and it shall not become hotter than t_{big}. Exceedings of t_{max} are to be avoided absolutely. Therefore such exceedings are evaluated by a penalty function, which reduces the total fitness achieved by further criteria by its penalty factor.

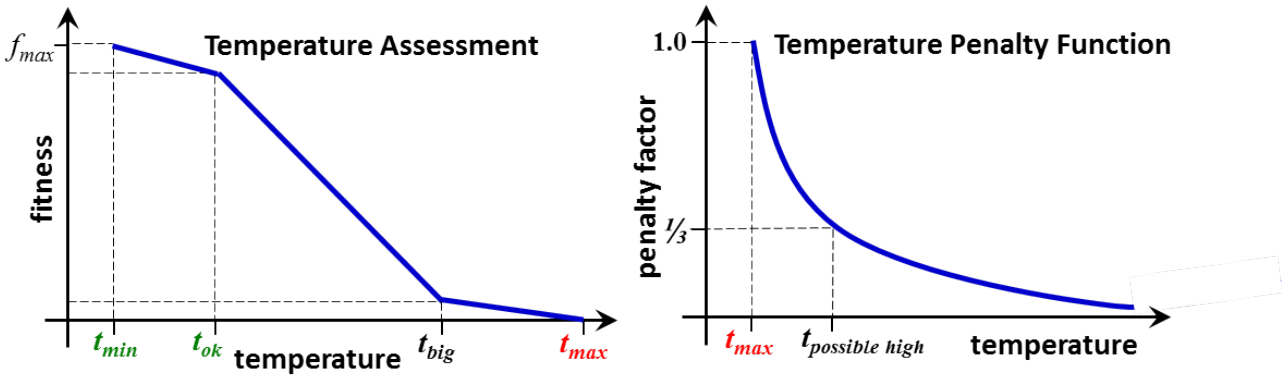

Fig. 4.1: Evaluation function for assessing a temperature including penalty function for exceeding it

1: The latter are in the field `simu_erg` from module `bewert.c` in the package `bew`, which is also used in the module `simu.c` of the same named package `simu`.

First the weight of the existing criterion is reduced to 60% by calling the menu item "ChangeCrit":

```
------------------------ Evaluation ------------------------
1: Show Crit.   2: ChangeCrit   3: New Crit.
5: ShowEvalF    6: ChangeEval   7: PlotEvalF
9: ShowPenFct   A: ChangePenF   B: PlotPenFct
D: Plot->File   E: Quit Plots   F: Close
------------------------------------------------------------
Select menu item: 2

                  Modification of a Criterion:
Nr. Criterion            Prio.    Fitness   Fulfillment     Penalty Fmax
                         Class      Share  W.Fitn   Value PrioClass Cont
 1: Griewank fct value :   1       100000   99999  0.0001  inactive  yes
------------------------------------------------------------------------
                                   100000
                                   ======
Select Criterion (1.. 1): 1

                         max. length: xxxxxxxxxxxxxxxxxx
New designation [Griewank fct value]: ↵
          max. length: xxxxxxxxxxxx
New scale unit        : ↵
With alternative measuring unit [y/N]: ↵
Priority class (0=deactivation)  [ 1]: ↵
  Remaining fitness value: 0
Fitness share    (0..100000) [100000]: 60000
Fulfillment value          [  0.0001]: ↵
PenaltyFuncPrio (0=Off)          [ 0]: ↵
Maintain max fitness in case of x-limit surpassing [Y/n]: ↵
  Remaining fitness value: 40000
**** Error: Fitness sum = 60000 instead of 100000!
```

The already existing values of the criterion are displayed as default, which can be taken over with <return>. Note that the name of a criterion may contain spaces, as the first dialog shows. With the green highlighted dialog "Fitness share" the value is lowered to 60000. When all dialogs have been completed, a note is displayed indicating the fitness portions that are currently not assigned ("Remaining fitness value") followed by a corresponding error message. This is intended to prompt the user to assign the weights or fitness portions completely. For this purpose now a new criterion is defined by "New Crit.":

```
------------------------ Evaluation ------------------------
1: Show Crit.   2: ChangeCrit   3: New Crit.
5: ShowEvalF    6: ChangeEval   7: PlotEvalF
9: ShowPenFct   A: ChangePenF   B: PlotPenFct
D: Plot->File   E: Quit Plots   F: Close
------------------------------------------------------------
Select menu item: 3

                  Definition of a new Criterion:
        max. length: xxxxxxxxxxxxxxxxxx
New designation [ ]: Temperature
          max. length: xxxxxxxxxxxx
New scale unit        : degs C
With alternative measuring unit [Y/n]: n
Priority class (0=deactivation)  [ 0]: 1
  Remaining fitness value: 40000
Fitness share    (0..100000) [     0]: 40000
```

```
A fulfillment value can be specified only after the definition of the re-
lated fitness function.
PrioClass of penalty fct.(0=Off) [ 0]: 1
Maintain max fitness in case of x-limit surpassing [Y/n]: ↵
  Remaining fitness value: 0

Nr. Criterion            Prio.    Fitness   Fulfillment     Penalty Fmax
                         Class      Share  W.Fitn   Value PrioClass Cont
 1: Griewank fct value :   1        60000   59999  0.0001  inactive  yes
 2: Temperature        :   1        40000   yet undefined     1      yes
-------------------------------------------------------------------------
                                   100000
                                   ======
```

Note that in a line above the name and unit of measurement dialogs for the new criterion, the respective maximum length is specified by an "x" string.

The dialog "With alternative measuring unit" allows to switch to the next larger or smaller thousand scale when displaying values. For example, 12523 m can be displayed as 12,523 km, if the appropriate settings are made.

The dialog „Priority class“ defines the priority class of the new criterion, with 1 being the highest.

Next, the remaining fitness portion is allocated. Afterwards it is pointed out that after the specification of an evaluation function a suitable fulfillment value is still to be assigned.

"PrioClass of penalty fct." is set to 1 and thus a penalty function yet to be defined is activated.

In the following dialog "Maintain max fitness in case of x-limit surpassing" the default value ("yes") is accepted.

Finally, the remaining fitness value is shown and the new criteria list is displayed.

Next, the evaluation function shown in Figure 4.1 must be specified. The following temperatures are assumed as an example: $t_{min}=40, t_{ok}=65, t_{big}=120$ and $t_{max}=150$. As fitness values for the two temperaturesn t_{ok} and t_{big} 95000 and 10000 are chosen. This is specified in the following dialog of menu item “ChangeEval”:

```
------------------------ Evaluation ------------------------
1: Show Crit.   2: ChangeCrit   3: New Crit.
5: ShowEvalF    6: ChangeEval   7: PlotEvalF
9: ShowPenFct   A: ChangePenF   B: PlotPenFct
D: Plot->File   E: Quit Plots   F: Close
------------------------------------------------------------
Select menu item: 6

      Modification of the Assessment Function of a Criterion:
Select Criterion (1.. 2) [ 2]: ↵

Assessment function for criterion "Temperature":

Function type: 1=IncrLin, 3=IncrExp, 5=IncrLinExp,
               2=DecrLin, 4=DecrExp, 6=DecrLinExp
Function type: 2
type of function : monotonically decreasing, linear
x-minimum [          0]: 40
x-point1             : 65
x-point2             : 120
x-maximum            : 150

Fitness values (0 .. 100000):
FcnPoint1 [          0]: 95000
FcnPoint2 [          0]: 10000
```

In the first dialog, the previously newly defined criterion is offered as default.

When the function type is subsequently selected, falling and linear is specified, which is confirmed in the following line by "monotonically decreasing, linear".

This is followed by the specifications for the four temperature values on the x-axis and then the fitness values for the two support points.

The specifications are next checked by plotting the function just defined:

```
------------------------ Evaluation ------------------------
1: Show Crit.    2: ChangeCrit    3: New Crit.
5: ShowEvalF     6: ChangeEval    7: PlotEvalF
9: ShowPenFct    A: ChangePenF    B: PlotPenFct
D: Plot->File    E: Quit Plots    F: Close
------------------------------------------------------------
Select menu item: 7

          Test of the Assessment Function of a Criterion:
Select Criterion (1.. 2) [ 2]: ↵
number of x values         [201]: ↵
x start value      [    40.00   ]: ↵
delta of x values [        0.55]: ↵
```

As you can see, all offered default values are used here. With the first three you can change the section to be plotted, which meanwhile can also be done using the functions of the GnuPlot window. The default values are calculated to give a good overview of the function.

With the help of the last dialog, the plot window can be determined, whereby up to 6 windows can be used in parallel. Figure 4.2 shows the evaluation function as GnuPlot.

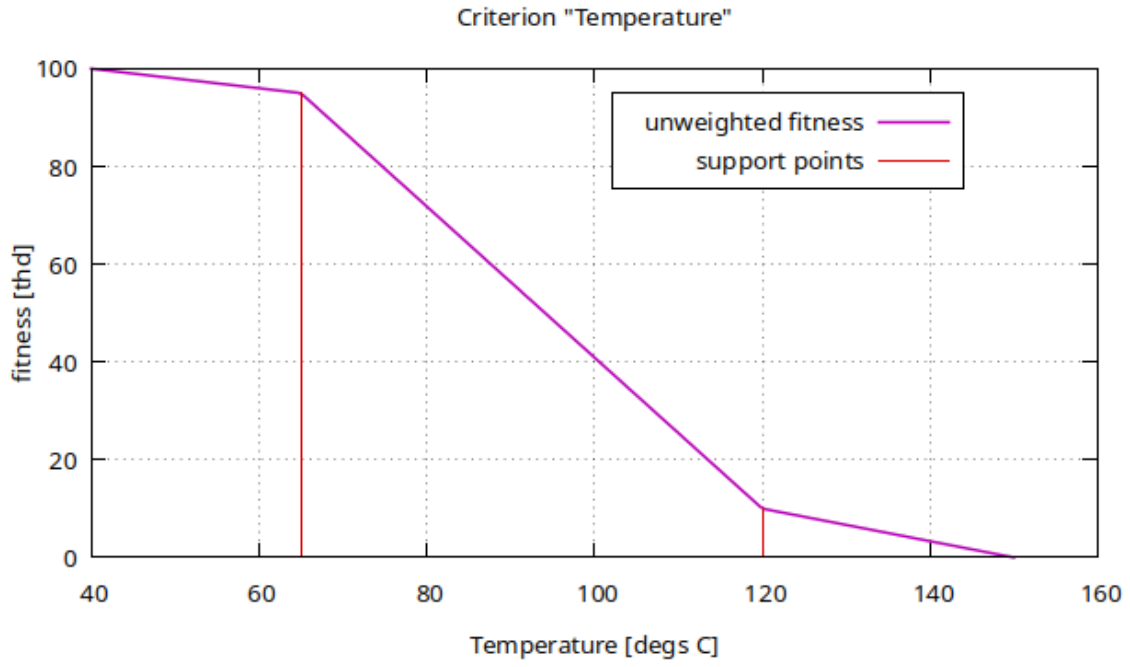

Fig. 4.2: Plot of the previously defined function for evaluating a temperature value

Still missing is the fulfillment value, which must lie within the interval of the temperature values recorded by the evaluation function. As an example it shall be set to the value 78°C:

```
------------------------ Evaluation ------------------------
1: Show Crit.    2: ChangeCrit    3: New Crit.
5: ShowEvalF     6: ChangeEval    7: PlotEvalF
9: ShowPenFct    A: ChangePenF    B: PlotPenFct
```

```
D: Plot->File   E: Quit Plots   F: Close
-----------------------------------------------------------------
Select menu item: 2

                  Modification of a Criterion:
Nr. Criterion            Prio.    Fitness   Fulfillment     Penalty Fmax
                         Class      Share  W.Fitn   Value PrioClass Cont
 1: Griewank fct value :   1        60000   59999  0.0001  inactive  yes
 2: Temperature        :   1        40000   yet undefined     1      yes
-------------------------------------------------------------------------
                                   100000
                                   ======
Select Criterion (1.. 2) [ 2]: ↵

                 max. length: xxxxxxxxxxxxxxxxxx
New designation [Temperature]: ↵
           max. length: xxxxxxxxxxxx
New scale unit [degs C]: ↵
With alternative measuring unit [y/N]: ↵
Priority class (0=deactivation)  [ 1]: ↵
  Remaining fitness value: 0
Fitness share    (0..100000) [ 40000]: ↵
Fulfillment value           [       0]: 78
PrioClass of penalty fct.(0=Off) [ 1]: ↵
Maintain max fitness in case of x-limit surpassing [Y/n]: ↵
  Remaining fitness value: 0
```

After selecting the menu item "ChangeCrit" the criteria list is displayed and the criterion to be changed is selected. All entries remain unchanged in the example except for the green marked "Fulfillment value".

A subsequent plot results in Figure 4.3 with the fulfillment value entered.

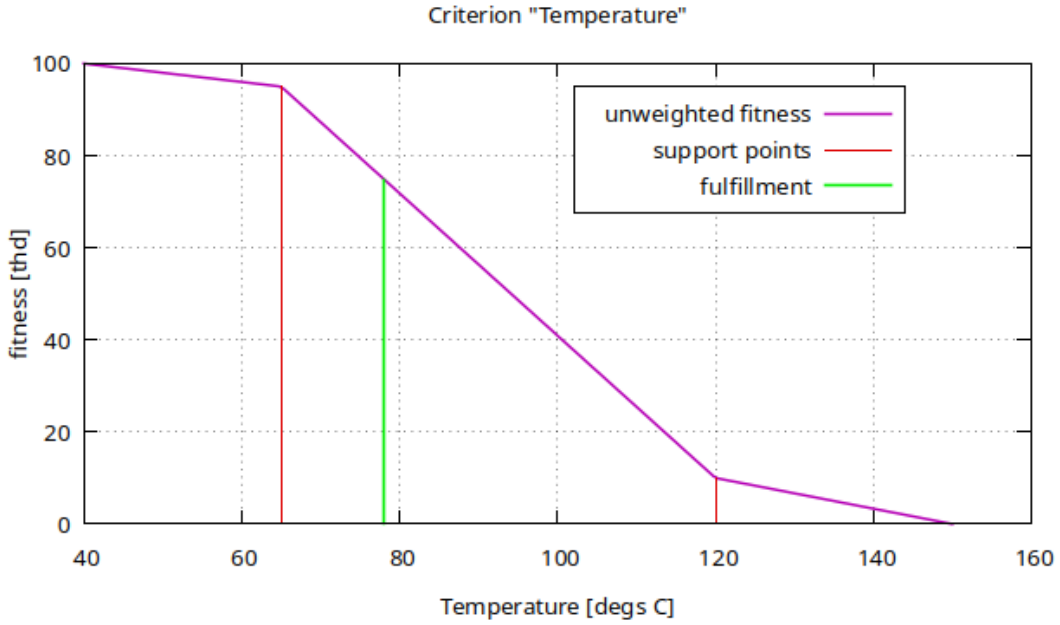

Fig 4.3: Plot of the function to evaluate a temperature including the fulfillment value

This completes the definition of an evaluation or normalization function for the new criterion and the definition of the penalty function shown in Figure 4.1 follows. For this purpose an exponential function is usually used, because it can also capture extremely large (or small) values and usually it is not known a priori to what extent a limit, as here t_{max}, can be exceeded. As an example 200 is chosen as value for $t_{possible\ high}$ and assigned to the value "x-third". The function value ⅓ is assigned to the

value "x-third" and defines the exponential function together with the minimum (or maximum for increasing functions). $t_{possible\ high}$ can be regarded as a realistic but too high value. The purpose of the penalty function is to show the evolutionary search a way out of the forbidden zone by penalizing lower temperature transgressions less than higher ones. Insofar $t_{possible\ high}$ can be chosen to fulfill this purpose.

```
------------------------ Evaluation ------------------------
1: Show Crit.   2: ChangeCrit   3: New Crit.
5: ShowEvalF    6: ChangeEval   7: PlotEvalF
9: ShowPenFct   A: ChangePenF   B: PlotPenFct
D: Plot->File   E: Quit Plots   F: Close
------------------------------------------------------------
Select menu item: a

       Modification of the Penalty Function of a Criterion:
Select Criterion (1.. 2) [ 2]: ↵

Penalty function for criterion "Temperature":

Function type: 1=IncrLin, 3=IncrExp, 5=IncrLinExp,
               2=DecrLin, 4=DecrExp, 6=DecrLinExp
Function type: 4
type of function : monotonically decreasing, exponential
x-minimum [          0]: 150
x-third                : 200
```

It should be noted that a penalty function only makes sense as a supplement to an evaluation function if it covers the value range from which fitness is zero, which in this example means values from 150. GLEAM does not verify this and it is the responsibility of the user to combine evaluation and penalty functions in a meaningful way.

It can also be useful to implement a restriction as a pure penalty function. For this purpose, the weight of the associated criterion has to be set to zero and the evaluation function has to be defined in such a way that it only captures values that do not occur or only those that do not violate the restriction.

To check the definition of the penalty function, it is also plotted using the menu item "PlotPenFct" and its following dialog:

```
------------------------ Evaluation ------------------------
1: Show Crit.   2: ChangeCrit   3: New Crit.
5: ShowEvalF    6: ChangeEval   7: PlotEvalF
9: ShowPenFct   A: ChangePenF   B: PlotPenFct
D: Plot->File   E: Quit Plots   F: Close
------------------------------------------------------------
Select menu item: b

          Test of the Penalty Function of a Criterion:

Select Criterion (1.. 2) [ 2]: ↵
number of x values        [201]: ↵
x start value     [   150.0   ]: ↵
delta of x values [       0.75]: ↵
plot window number (1..6)   [1]: 2
```

This results in the penalty function shown in Figure 4.4 in a second window of GnuPlot.

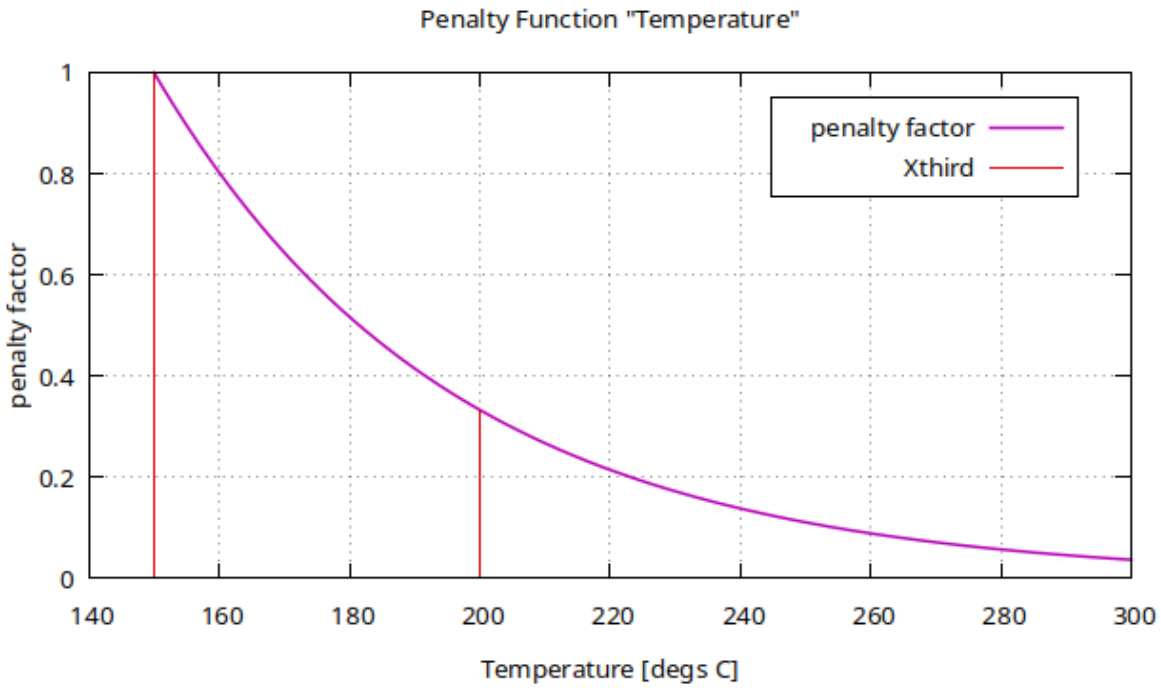

Fig 4.4: Penalty function for too high temperatures

Finally, the list of the now defined criteria is displayed using the menu item "Show Crit.":

```
                        Assessment Criteria:

Nr. Criterion            Prio.    Fitness   Fulfillment     Penalty Fmax
                         Class      Share  W.Fitn   Value PrioClass Cont
 1: Griewank fct value :   1        60000   59999  0.0001  inactive  yes
 2: Temperature        :   1        40000   29964      78     1      yes
-------------------------------------------------------------------------
                                   100000
                                   ======
```

4.2.7 Simulator Menu

In this context, "simulator" means everything that serves for interpretation and evaluation of a chromosome, and provides values for the individual criteria as a result, from which a normalized fitness is calculated by the weighted cascaded sum (see menu Evaluation, Sect. 4.2.6). The fitness is a real value in the range between 0 and 100,000, where maximization is used.

The simulator menu is used to assign the (external) simulator to simulate a single chromosome and to present the results. In GLEAM, already simulated chromosomes contain the result values of the previous simulation, if there are less than 16[1]. If such a chromosome is simulated, the simulator accesses the result data stored in the chromosome, depending on the application specified in the MOD file, and only a new calculation of the cascaded weighted sum is performed. This procedure serves to avoid the usually time-consuming simulations and is intended for cases where only a modified evaluation is to be tested. To actually call the simulator, the old values have to be deleted before. For this purpose, the menu item "DelSimVal" deletes the evaluation data of a single chromosome, and the menu item "MemEvalUpd" allows to simulate and re-evaluate the entire chromosome memory in one step.

1: Value of AK_ERG_WERTE_MAX in the general configuration file "chaindef.h" in the root directory of the "sources".

To facilitate chromosome selection, the list of chromosomes stored per fitness class is displayed when "SimuStart" is called:

```
------------------------ Simulator --------------------------
1: SimuStart
                                7: State
9: Log.on/off                   B: DelSimVal    C: MemEvalUpd
D: Close
-------------------------------------------------------------
Select menu item: 1

    Simulation of a Chromosome:
  Class    Fitness Interval  Number
     1          0 -      0      0
     2          1 -   2499      0
     3       2500 -   4999      0
     4       5000 -   9999      1
     5      10000 -  19999      0
     6      20000 -  29999      0
     7      30000 -  39999      0
     8      40000 -  49999      0
     9      50000 -  59999      0
    10      60000 -  69999      1
    11      70000 -  74999      1
    12      75000 -  79999      0
    13      80000 -  84999      0
    14      85000 -  89999      0
    15      90000 -  94999      1
    16      95000 - 100000      1
Chr [class/index]: 15
Chr [class/index]: 15/ 1

              Simulation Results for Chromosome 15/1:
Fitness remains unchanged: 91354 (91353.958106919)
```

Since the evaluation was not changed, the result also corresponds to the fitness stored in the chromosome. The simulation changes the Simulator menu and menu items become available for a more detailed display of the result:

```
------------------------ Simulator --------------------------
1: SimuStart    2: ShowEval     3: PlotEval     4: SaveSimRes
5: EvoBest                      7: State
9: Log.on/off                   B: DelSimVal    C: MemEvalUpd
D: Close
-------------------------------------------------------------
Select menu item: 2

                    Evaluation of the Recent Simulation:
                  X-Values:
   p01                 : 15.87527
   p02                 : -18.43264
   p03                 : -33.80851
   p04                 : 5.952202
   p05                 : 28.17107
-------------------------------------------------------------
No. Objective           Prio Wght Unwght. --- Weighted ---  Simulator Results
                              [%] Fitness MinFitn. Fitness
 1: Griewank fct value :   1  100   91354    99999   91354   0.9606722
-------------------------------------------------------------
Total fitness (without penalty functions):          91354
```

```
No penalty functions applied.
Total fitness:                                          91354
                                                       ======
```

Depending on the application, the decision variables are displayed first, whereby parameter names and possible units of measurement are taken from the MOD file, see [7]. The table of criteria with the current values follows. In the right column the result values of the simulation are shown. Similar to the display of the simulation results the measurement units specified in the "Evaluation" menu are used. To demonstrate this, the detailed display of an optimization result for the robot task [2, 3, 6] is shown below:

```
------------------------ Simulator --------------------------
1: SimuStart
5: EvoBest                    7: State
9: Log.on/off                 B: DelSimVal    C: MemEvalUpd
D: Close
-------------------------------------------------------------
Select menu item: 5

       Evaluation of the Best Solution of the Actual Optimization Run:
-----------------------------------------------------------
No. Objective           Prio Wght Unwght. --- Weighted ---  Simulator Results
                           [%] Fitness MinFitn. Fitness
 1: target deviation   :   1   30   96910     8027   29073     0.72 %
 3: path deviation     :   1   50   99508     3189   49754    10.10 %
 4: action chain length:   2    7   99900     3780    6993       31 actions
 6: crash residual path:   2    0  100000        1       1        0 cm
 2: motion duration    :   3   10   94007     6623    9401      4.9 sec
 5: no of exec. instr. :   4    3  100000     2339    2999       30 instruc-
                                                                   tions
-----------------------------------------------------------
Total fitness (without penalty functions):           98221
No penalty functions applied.
Total fitness:                                       98221
                                                    ======
```

The "PlotEval" menu item can be used to create a plot of the evaluation or penalty function used depending on the result current value. The menu item uses dialogs similar to those used for plotting an evaluation or penalty function (see 4.2.6). At the end it is asked whether a Hard-copy shall be created. This option originates from the time when the GnuPlot program did not provide such functions yet. It is recommended to use these newer GnuPlot functions.

The menu item "SaveSimRes" creates a text file containing the above output.

The menu item "MemEvalUpd" allows to recalculate the fitness of all chromosomes in memory with and without re-simulation:

```
------------------------ Simulator --------------------------
1: SimuStart    2: ShowEval     3: PlotEval     4: SaveSimRes
5: EvoBest                    7: State
9: Log.on/off                 B: DelSimVal    C: MemEvalUpd
D: Close
-------------------------------------------------------------
Select menu item: c

   Update of the Evaluation of the Entire Chromosome Memory
        Based on Present Task and Evaluation Criteria
Deletion of the stored simulator results? [Y/n]:

Evaluation of 5 chromosomes will be updated ...
Evaluation of 5 chromosomes updated.
```

During or after an optimization run the "Simulator" menu offers the menu item "EvoBest".[1] This produces an output similar to the display of the results of a simulation. The biggest difference is the changed heading:

```
------------------------ Simulator --------------------------
1: SimuStart    2: ShowEval     3: PlotEval     4: SaveSimRes
5: EvoBest                      7: State
9: Log.on/off                   B: DelSimVal    C: MemEvalUpd
D: Close
-------------------------------------------------------------
Select menu item: 5

       Evaluation of the Best Solution of the Actual Optimization Run:
                  X-Values:
   p01                 : -0.0007203778
   p02                 : -0.0002945434
   p03                 : -0.0006144894
   p04                 : 0.0002639224
   p05                 : -0.001485355
-------------------------------------------------------------
No. Objective           Prio Wght Unwght. --- Weighted ---  Simulator Results
                              [%] Fitness MinFitn. Fitness
 1: Griewank fct value :   1  100  100000    99999  100000   5.742434E-07
-------------------------------------------------------------
Total fitness (without penalty functions):          100000
No penalty functions applied.
Total fitness:                                      100000
                                                    ======
```

The output of the simulator status of the menu item "State" is mainly required when external simulators are integrated. Depending on the connection it contains information about the communication status, the status of the simulator and the loaded model. For the internal simulator the output is accordingly lean:

```
------------------------ Simulator --------------------------
1: SimuStart    2: ShowEval     3: PlotEval     4: SaveSimRes
5: EvoBest                      7: State
9: Log.on/off                   B: DelSimVal    C: MemEvalUpd
D: Close
-------------------------------------------------------------
Select menu item: 7

                Simulator State:

Internal simulator ready    (sim_up) = active
Simulator logging                    = off
```

4.2.8 Evo/Opt Menu

The menu display is based on HyGLEAM. At first an output of the current parameter settings is done using "ShowEvoPar". Regarding the meaning of most parameters, please refer to Sect. 2.2.

```
------------------------ Evo/Opt ----------------------------
1: DefOptJob    2: ModOptJob    3: StartJobLi
5: ShowJobLi    6: Copy Job     7: Delete Job
9: SaveResChr   A: SavBestRes   B: ShowEvoPar   C: Opt Params
                E: Del Popul.   F: Close
-------------------------------------------------------------
Select menu item: b
```

1: It has been observed that sometimes not the best result of an optimization run is displayed. This is a known error.

```
       Present Parameters of Evolution (GLEAM):
       ========================================

Deme size                              :  8
Acceptance rule for offspring          : local least (elitist)
Ranking parameter (1.0 .. 2.0)         :  1.50
Minimum Hamming distance for Reco/XO   :  0.1 %
Application-specific repair procedures: -
Default LHC for start population init.: Rosenbrock procedure

Local Hill Climbers (LHC) according to gene model:
Their control parameters can be overwritten by the TSK-file.
  Rosenbrock procedure, control parameter:
    R-iteration limit           : 5000
    termination limit           : 1e-06
  Complex algorithm, control parameter:
    C-iteration limit           : 500
```

In the pure GLEAM application all information about local search methods (LHCs) would be missing. But if in HyGLEAM the EA GLEAM is selected as optimization method, implemented LHCs can play a role in so far as they can be used to initialize the initial population. Therefore, their current parameterization is displayed after the first five evolution-related parameters.

Based on this, an evolution job is to be defined, in which the three best chromosomes from the chromosome memory are used to seed the initial population. The best individual in the memory has a fitness of 57417:

```
------------------------- Evo/Opt --------------------------
1: DefOptJob    2: ModOptJob    3: StartJobLi
5: ShowJobLi    6: Copy Job     7: Delete Job
9: SaveResChr   A: SavBestRes   B: ShowEvoPar   C: Opt Params
                E: Del Popul.   F: Close
------------------------------------------------------------
Select menu item: 1

                  Define Optimization Job 1:

OptimProc:  GLEAM  SMA  ASMA  M=AMMA  L=LHC       [G]: ↵
Population size                    ( 8..20000) [ 300]: ↵
Init: New  Best  Mix  +=B+N  LHC  #=LHC+B  File   [N]: +
Strategy parameter: number of best chromosomes  [  0]: 3
Initialise random number generator (RNG) ?      [Y/n]: ↵
Acceptance rule: 1=LL 2=Alw 3=LLES 4=AlwES 5=BP   [3]: ↵
Ranking parameter                   (1.0..2.0) [1.50]: ↵
Minimal HammingDist for recomb./x-over in % [    0.1]: ↵
Population backup rate (generations)     [2147483647]: ↵
(Max) number of result chromosomes (0.. 150)   [   1]: ↵

                  Termination Conditions:
Minimum fitness                                [100000]: ↵
Maximum number of generations                   [10000]: 12
Max. gen. without deme improvement (GDI)        [   12]: ↵
Max. gen. without offspring acceptance (GAc)    [   12]: ↵
Maximum number of evaluations              [2147483647]: ↵
Maximum job time in minutes   [00:04:00 =      4 min]: ↵
```

As you can see, most of the default values are used. But this job is only for demonstration purposes and should therefore only run for a short time, here 12 generations. Accordingly the default values for GDI and GAc are reduced to this small value. The job list display "ShowJobLi" shows the following according to the input:

```
------------------------- Evo/Opt --------------------------
1: DefOptJob    2: ModOptJob    3: StartJobLi
5: ShowJobLi    6: Copy Job     7: Delete Job
9: SaveResChr   A: SavBestRes   B: ShowEvoPar   C: Opt Params
                E: Del Popul.   F: Close
------------------------------------------------------------
Select menu item: 5

                          Job List:

---Job-- --Sizes- -------Init------- Res -----------Target Values-----------
 No OptP Pop/Deme  RNG   Popul  Spar Chr Fitness hh:mm   Gen  GDI GAc MaxEval
------------------------------------------------------------------------------
  1  Evo   300/8   yes bestNew     3   1  100000  0:04    12   12  12       -
```

Menu item "StartJobLi" starts the job (and others if the list would contain more jobs):

```
------------------------- Evo/Opt --------------------------
1: DefOptJob    2: ModOptJob    3: StartJobLi
5: ShowJobLi    6: Copy Job     7: Delete Job
9: SaveResChr   A: SavBestRes   B: ShowEvoPar   C: Opt Params
                E: Del Popul.   F: Close
------------------------------------------------------------
Select menu item: 3

                  Start Optimization Job List:

---Job-- --Sizes- -------Init------- Res HamDist Rank AcceptanceRule
 No OptP Pop/Deme  RNG   Popul  Spar Chr     [%]  Par  for Offspring
  1  Evo   300/8   yes bestNew     3   1     0.1  1.5  localLeast-ES
-----------------------------------------------+--------------------
       Fitness  Runtime   Gen  GDI GAc MaxEval |
 Limit: 100000  0:04:00    12   12  12       - |
-----------------------------------------------+
  Init:  57417  0:00:00     0    0   0     297 |
ActVal:  63485  0:00:00     1    0   0    1565 |
ActVal:  64600  0:00:00     2    0   0    2878 |
ActVal:  64600  0:00:00     3    0   0    4139 |
ActVal:  81899  0:00:00     4    0   0    5429 |
ActVal:  82915  0:00:00     5    0   0    6697 |
ActVal:  86334  0:00:00     6    0   0    7984 |
ActVal:  86832  0:00:00     7    0   0    9240 |
ActVal:  91386  0:00:00     8    0   0   10475 |
ActVal:  93300  0:00:00     9    0   0   11738 |
-----------------------------------------------+
       Fitness  Runtime   Gen  GDI GAc MaxEval |
 Limit: 100000  0:04:00    12   12  12       - |
-----------------------------------------------+
ActVal:  95615  0:00:00    10    0   0   12941 |
ActVal:  95615  0:00:00    11    0   0   14195 |
ActVal:  95615  0:00:00    12    0   0   15391 |
-----------------------------------------------+--------------------------------
       Fitness  Runtime   Gen  GDI GAc MaxEval |         Job List Results
 Limit: 100000  0:04:00    12   12  12       - | Number BestChrom.: Fitness Length
ActVal:  95615  0:00:00    12    0   0   15391 |      1               95615      5
-----------------------------------------------+--------------------------------
```

The first lines document the job parameters. The line "Init:" reflects the situation after initialization of the initial population: Only 297 new chromosomes were generated and evaluated, since the remaining 3 were taken from the chromosome memory. The best of these chromosomes is the one with a fitness of 57417. Then the evolution begins and after each generation there is an output. After 10 generations the column headings and the target values are repeated. If one of the termination criteria (here the 12 generations) is reached, the result is displayed.

The evolutionary process can be interrupted by any input. But you should consider, that this input is interpreted too and therefore you should enter something "meaningful". With ESC for example you get into the main menu. From there you can enter the “Simulator” menu, where you can see e.g. the rating of the current best individual by selecting "EvoBest".[1] If you remain in the Evo/Opt menu, you can use "Delete job" to delete e.g. waiting jobs from the list or the current one, whereby you can then specify whether the current generation should still be completed, which includes the saving of the results, or whether the job should be terminated immediately. A further alternative is to modify waiting jobs or the active one. In the latter case only termination parameters can be modified.

With the menu item "Opt Params" you can change the current optimization procedure and the respective (default) parameters. The listing of the current values is partly omitted in the following for reasons of overview. It is assumed that the EA GLEAM is the default optimization procedure when entering the “Opt Params” menu.

```
------------------------- Evo/Opt --------------------------
1: DefOptJob    2: ModOptJob    3: StartJobLi
5: ShowJobLi    6: Copy Job     7: Delete Job
9: SaveResChr   A: SavBestRes   B: ShowEvoPar   C: Opt Params
                E: Del Popul.   F: Close
------------------------------------------------------------
Select menu item: c

      Default Settings for Optimization Parameters:

        Present Parameters of Evolution (GLEAM):
        ========================================

Deme size                           :  8
Acceptance rule for offspring       : local least (elitist)
Ranking parameter (1.0 .. 2.0)      :  1.50
Minimum Hamming distance for Reco/XO  :  0.1 %
Application-specific repair procedures: -
Default LHC for start population init.: Rosenbrock procedure

Local Hill Climbers (LHC) according to gene model:
Their control parameters can be overwritten by the TSK-file.
  Rosenbrock procedure, control parameter:
    R-iteration limit           : 5000
    termination limit           : 1e-06
  Complex algorithm, control parameter:
    C-iteration limit           : 500
```

First, the actual parameters of the currently set optimization method are displayed. Some of these parameters can be changed when defining an optimization job.

Next, the optimization method Simple Memetic Algorithm (SMA), i.e. a simple memetic algorithm with one LHC and without adaptation, is set as default.

```
------------------------ Opt Params ------------------------
1: OptStrat                     3: Deme Size    4: AcceptRule
5: Rank-Param   6: HammDistXO
                A: select LS    B: LHC Params   C: Close
------------------------------------------------------------
Select menu item: 1

           Selection of a New Optimization Procedure

Present default optimization procedure : GLEAM
```

1: It has been observed that sometimes not the best result of an optimization run is displayed. This is a known error.

```
------------------------- OptStrat -------------------------
1: GLEAM        2: GLEAM-SMA    3: GLEAM-ASMA   4: GLEAM-AMMA
5: LHC          6: Close
------------------------------------------------------------
Select menu item: 2

        Present Parameters of Evolution (GLEAM-SMA):
        ============================================
                 Display of the same parameters as above
```

With this the simple memetic algorithm with GLEAM and the Rosenbrock method is set as meme. The "Opt Params" menu now shows the new menu item "LamarckPar" which allows to switch between Lamarckian and Baldwinian evolution (with or without updating the chromosome according to the LHC result). The parameter allows a percentage distribution between both approaches, where 100% corresponds to Lamarckian evolution and 0% to pure Baldwin evolution. In the example, the respective chromosome is adjusted at 75% of the improved results:

```
------------------------ Opt Params ------------------------
1: OptStrat                     3: Deme Size    4: AcceptRule
5: Rank-Param   6: HammDistXO
9: LamarckPar   A: Select LS    B: LHC Params   C: Close
------------------------------------------------------------
Select menu item: 9

Lamarck rate (chromosome update rate in case of LHC success) in % [100]: 75

        Present Parameters of Evolution (GLEAM-SMA):
        ============================================
             Similar display of the same parameters as above
```

Using the "LHC Params" menu item you can change the parameters of the current LHC:

```
------------------------ Opt Params ------------------------
1: OptStrat                     3: Deme Size    4: AcceptRule
5: Rank-Param   6: HammDistXO
9: LamarckPar   A: Select LS    B: LHC Params   C: Close
------------------------------------------------------------
Select menu item: b

Parameterization of LHCs "Rosenbrock procedure":
  R-iteration limit                   [   5000]: 3333
  termination limit                   [  1e-06]: 5e-05

        Present Parameters of Evolution (GLEAM-SMA):
        ============================================

Deme size                           :  8
Acceptance rule for offspring       : local least (elitist)
Ranking parameter (1.0 .. 2.0)      :  1.50
Minimum Hamming distance for Reco/XO  :  0.1 %
Application-specific repair procedures: -
Lamarck or chromosome update rate   : 75.0 %
Present default Local Hill Climber LHC: Rosenbrock procedure

Local Hill Climbers (LHC) according to gene model:
Their control parameters can be overwritten by the TSK-file.
  Rosenbrock procedure, control parameter:
    R-iteration limit           : 3333
    termination limit           : 5e-05
  Complex algorithm, control parameter:
    C-iteration limit           : 500
```

To change the iteration limit of the Complex method, you first have to select it and then change its strategy parameter. Finally, the ranking parameter should be increased and the acceptance rule should be changed:

```
------------------------ Opt Params -------------------------
1: OptStrat                    3: Deme Size    4: AcceptRule
5: Rank-Param   6: HammDistXO
9: LamarckPar   A: Select LS    B: LHC Params   C: Close
-------------------------------------------------------------
Select menu item: 5

Default ranking param. of mate selection (1..2) [1.50]: 1.4

        Present Parameters of Evolution (GLEAM-SMA):
        ============================================
             Similar display of the same parameters as above

------------------------ Opt Params -------------------------
1: OptStrat                    3: Deme Size    4: AcceptRule
5: Rank-Param   6: HammDistXO
9: LamarckPar   A: Select LS    B: LHC Params   C: Close
-------------------------------------------------------------
Select menu item: 4

           Selection of a New Default Acceptance Rule

Present default acceptance rule    : local least (elitist)

------------------------ AcceptRule -------------------------
1: BetParent    2: LocLeastES   3: AlwaysES
                6: LocalLeast   7: Always       8: Close
-------------------------------------------------------------
Select menu item: 1

        Present Parameters of Evolution (GLEAM-SMA):
        ============================================

Deme size                           :  8
Acceptance rule for offspring       : better parent
Ranking parameter (1.0 .. 2.0)      :  1.40
Minimum Hamming distance for Reco/XO  :  0.1 %
Application-specific repair procedures: -
Lamarck or chromosome update rate     :  75.0 %
Present default Local Hill Climber LHC: Rosenbrock procedure

Local Hill Climbers (LHC) according to gene model:
Their control parameters can be overwritten by the TSK-file.
  Rosenbrock procedure, control parameter:
    R-iteration limit           : 3333
    termination limit           : 5e-06
  Complex algorithm, control parameter:
    C-iteration limit           : 500
```

With these settings a new SMA-C job with a small population size is defined and started for a short run. Experience shows that the population size can generally be selected (much) smaller for the memetic extensions of the EA than for pure evolution.

```
------------------------ Evo/Opt ----------------------------
1: DefOptJob    2: ModOptJob    3: StartJobLi
5: ShowJobLi    6: Copy Job     7: Delete Job
```

```
9: SaveResChr   A: SavBestRes   B: ShowEvoPar   C: Opt Params
                E: Del Popul.   F: Close
------------------------------------------------------------
Select menu item: 1

                  Define Optimization Job 1:

OptimProc:  GLEAM  SMA  ASMA  M=AMMA  L=LHC       [S]: ↵
LHC: Rosenbrock ComplexAlg                        [R]: ↵
Population size                    ( 8..20000) [  20]: ↵
Init: New  Best  Mix  +=B+N  LHC  #=LHC+B  File   [N]: ↵
Strategy parameter: Chromosomes with fitn. > 0  [  0]: ↵
Initialise random number generator (RNG) ?      [Y/n]: ↵
Acceptance rule: 1=LL 2=Alw 3=LLES 4=AlwES 5=BP   [5]: ↵
Ranking parameter                   (1.0..2.0) [1.40]: ↵
Minimal HammingDist for recomb./x-over in % [    0.1]: ↵
Population backup rate (generations)     [2147483647]: ↵
(Adaptive) LHC-optimization of all offspring?   [y/N]: ↵
Chromosome update rate (Lamarck rate) in %     [  75]: ↵
(Max) number of result chromosomes (0..  10)   [   1]: ↵

                  Termination Conditions:
Minimum fitness                                [100000]: ↵
Maximum number of generations                   [10000]: 8
Max. gen. without deme improvement (GDI)        [    8]: ↵
Max. gen. without offspring acceptance (GAc)    [    8]: ↵
Maximum number of evaluations              [2147483647]: ↵
Maximum job time in minutes   [00:04:00 =      4 min]: ↵

-------------------------- Evo/Opt ---------------------------
1: DefOptJob    2: ModOptJob    3: StartJobLi
5: ShowJobLi    6: Copy Job     7: Delete Job
9: SaveResChr   A: SavBestRes   B: ShowEvoPar   C: Opt Params
                E: Del Popul.   F: Close
------------------------------------------------------------
Select menu item: 3

                  Start Optimization Job List:

---Job-- --Sizes- -------Init------- Res HamDist Rank AcceptanceRule  active
 No OptP Pop/Deme  RNG   Popul  Spar Chr     [%]  Par  for Offspring    LHCs
  1  SMA    20/8   yes     new     0   1     0.1  1.4  better parent     Ros
-----------------------------------------------+-------------------------
       Fitness  Runtime   Gen  GDI GAc MaxEval |
 Limit: 100000  0:04:00     8    8   8       - |
-----------------------------------------------+
  Init:   9680  0:00:00     0    0   0      20 |
ActVal:  99402  0:00:00     1    0   0    9368 |
ActVal:  99579  0:00:00     2    0   0   14901 |
ActVal:  99645  0:00:00     3    0   0   23732 |
ActVal:  99645  0:00:00     4    0   0   42022 |
ActVal:  99690  0:00:00     5    0   0   77636 |
ActVal:  99911  0:00:00     6    0   0  129355 |
ActVal:  99911  0:00:01     7    0   0  183746 |
ActVal:  99913  0:00:01     8    0   0  241135 |
-----------------------------------------------+---------------------------------
       Fitness  Runtime   Gen  GDI GAc MaxEval |         Job List Results
 Limit: 100000  0:04:00     8    8   8       - | Number BestChrom.: Fitness Length
ActVal:  99913  0:00:01     8    0   0  241135 |      1                99911      5
-----------------------------------------------+---------------------------------
```

Finally a run of the self-adaptive multimeme version of HyGLEAM is described. It is highly recommended to read the paper about this extension [8] first. To begin with we take a look at the “Opt Params” menu and the available parameters:

```
------------------------- Evo/Opt --------------------------
1: DefOptJob    2: ModOptJob    3: StartJobLi
5: ShowJobLi    6: Copy Job     7: Delete Job
9: SaveResChr   A: SavBestRes   B: ShowEvoPar   C: Opt Params
                E: Del Popul.   F: Close
------------------------------------------------------------
Select menu item: c

      Default Settings for Optimization Parameters:

        Present Parameters of Evolution (GLEAM-AMMA):
        =============================================

Deme size                              :  8
Acceptance rule for offspring          : local least (elitist)
Ranking parameter (1.0 .. 2.0)         :  1.50
Minimum Hamming distance for Reco/XO   :  0.1 %
Application-specific repair procedures: -
Lamarck or chromosome update rate      : 100.0 %

Interactive adjustable control parameters of adaptation:
Initial probabilities of the LHCs: R=0.5 C=0.5
Values of the LHC and level adaptation speeds "fast" and "fast":
  Minimum applications of one/all LHCs           : 3 / 15
  Minimum applications of one/all levels         : 3 / 12
Number of fitness classes with (limits, in thsd) : 3  (40, 75, 100)
Fraction of the old LHC distrib. for the new one : 33.333 %
Small Plhc repetition limit for LHC deactiviation: 3
Initial Pmin for LHC deactiviation               : 10.0 %

Gene model included control parameters of the adaptation:
Values of the 6 levels for all-P:  0 0.2 0.4 0.6 0.8  1
LHC "Rosenbrock procedure":
    all-P: Startlevel and initial probabilities: 0  (33.4, 33.3, 33.3)
    LHC parameter "R-iteration limit":
      Startlevel and initial probab.: 0  (50, 30, 20)
      Values of the 10 levels:  100 200 350 500 750  1000 1250 1500 1750 2000
    LHC parameter "termination limit":
      Startlevel and initial probab.: 0  (33.4, 33.3, 33.3)
      Values of the 9 levels:  0.1 0.01 0.001 0.0001 1e-05 1e-06 1e-07 1e-08 1e-09
LHC "Complex algorithm":
    all-P: Startlevel and initial probabilities: 0  (33.4, 33.3, 33.3)
    LHC parameter "C-iteration limit":
      Startlevel and initial probab.: 0  (50, 30, 20)
      Values of the 10 levels:  50 100 150 200 300  400 500 650 800 1000

------------------------ Opt Params ------------------------
1: OptStrat                    3: Deme Size    4: AcceptRule
5: Rank-Param   6: HammDistXO
9: LamarckPar   A: LSIniDistr   B: AdaptSpeed   C: FitnClass
D: AdaptOld     E: NoLHCLimit   F: Close
------------------------------------------------------------
```

The first parameter block contains the already known parameters of GLEAM and the Lamarck rate of the general MA extension. The second block consists of parameters that can be changed interactively by the menu items from “LSIniDistr” on. The parameters and their corresponding menu items

are listed in the same sequence. The situation shown above is based on the two LHCs “Rosenbrock procedure” with two control parameters and “Complex algorithm” with one parameter.

“LSIniDistr” allows to specify the initial probabilities of the LHCs involved.

With “AdaptSpeed” the speed of adaptation can be parameterized, see Table 2 in [8].

The adaptation can be done separately for different fitness ranges, see Sect. 4.4 of [8]. These ranges are called “fitness classes” here and they can be configured with “FitnClass”.

To soften changes in the probability distribution of the LHCs, a part of the old distribution can be included, see also Sect. 3.2 of [8] and there the description of the “extended meme adaptation”. This can be controlled by the “AdaptOld” menu function.

If the probability of an LHC falls below a threshold value several times in a row, it is switched off for this fitness class, see also Sect. 3.2 of [8]. The value of p_{min} is calculated depending on the number of active LHCs and can not be altered manually. It is set to 20% of the probability of an LHC, which would result with the distribution of all active LHCs being equal. “NoLHCLimit” allows to change the number of calculated probability underruns that occur in a sequence. The initial value of p_{min} is shown at the end of the second block.

The third block contains the levels for controlling the strategy parameters of the local searchers and the probabilities for “adaptive all improvement”, see [8] and in particular Table 4.1 and Figure 5.

The initial values of the parameters of the second block and those from the third one are contained in the MOD file, see [7].

Finally an optimization Job is defined using the default values and started:

```
------------------------- Evo/Opt --------------------------
1: DefOptJob    2: ModOptJob    3: StartJobLi
5: ShowJobLi    6: Copy Job     7: Delete Job
9: SaveResChr   A: SavBestRes   B: ShowEvoPar   C: Opt Params
                E: Del Popul.   F: Close
------------------------------------------------------------
Select menu item: 1


                  Define Optimization Job 1:

OptimProc:  GLEAM  SMA  ASMA  M=AMMA  L=LHC       [M]: ↵
Population size                    ( 8..20000) [  20]: ↵
Init: New  Best  Mix  +=B+N  LHC  #=LHC+B  File   [N]: ↵
Strategy parameter: Chromosomes with fitn. > 0  [  0]: ↵
Initialise random number generator (RNG) ?      [Y/n]: ↵
Acceptance rule: 1=LL 2=Alw 3=LLES 4=AlwES 5=BP   [3]: ↵
Ranking parameter                   (1.0..2.0) [1.50]: ↵
Minimal HammingDist for recomb./x-over in % [    0.1]: ↵
Population backup rate (generations)     [2147483647]: ↵
(Adaptive) LHC-optimization of all offspring?   [y/N]: ↵
Chromosome update rate (Lamarck rate) in %     [ 100]: ↵
(Max) number of result chromosomes (0..  10)   [   1]: ↵

                  Termination Conditions:
Minimum fitness                               [100000]: ↵
Maximum number of generations                  [10000]: ↵
Max. gen. without deme improvement (GDI)       [  999]: ↵
Max. gen. without offspring acceptance (GAc)   [  300]: ↵
Maximum number of evaluations             [2147483647]: ↵
Maximum job time in minutes   [00:04:00 =      4 min]: ↵
```

```
------------------------- Evo/Opt --------------------------
1: DefOptJob    2: ModOptJob    3: StartJobLi
5: ShowJobLi    6: Copy Job     7: Delete Job
9: SaveResChr   A: SavBestRes   B: ShowEvoPar   C: Opt Params
                E: Del Popul.   F: Close
------------------------------------------------------------
Select menu item: 3

                 Start Optimization Job List:

---Job-- --Sizes- -------Init------- Res HamDist Rank AcceptanceRule  active
 No OptP Pop/Deme  RNG   Popul  Spar Chr     [%]  Par  for Offspring    LHCs
  1 AMMA    20/8   yes     new     0   1     0.1  1.5  localLeast-ES      RC
-----------------------------------------------+--------------------------
       Fitness  Runtime   Gen  GDI GAc MaxEval |
 Limit: 100000  0:04:00 10000  999 300       - |
-----------------------------------------------+
  Init:   9843  0:00:00     0    0   0      20 |
ActVal:  99486  0:00:00     1    0   0    2138 |
ActVal:  99509  0:00:00     2    0   0    4082 |
ActVal:  99512  0:00:00     3    0   0    6477 |
ActVal:  99512  0:00:00     4    0   0    9169 |
ActVal:  99889  0:00:00     5    0   0   11443 |
ActVal:  99889  0:00:00     6    0   0   13786 |
ActVal:  99889  0:00:00     7    0   0   16547 |
ActVal: 100000  0:00:00     8    0   0   17915 |
-----------------------------------------------+----------------------------------
       Fitness  Runtime   Gen  GDI GAc MaxEval |         Job List Results
 Limit: 100000  0:04:00 10000  999 300       - | Number BestChrom.: Fitness Length
ActVal: 100000  0:00:00     8    0   0   17915 |      1              100000      5
-----------------------------------------------+----------------------------------
```

4.2.9 System Menu

The system menu contains functions for memory management, work with chromosomes and a number of test functions. They are explained in the order they appear in the menu.

```
------------------------- System ---------------------------
1: MemInfo      2: Create Chr   3: Check Chr    4: Chr-Editor
5: Del Chr      6: Del Chr/Cl   7: Del ChrMem
9: HamDist      A: MutOpRate    B: GenOp-Test   C: Gen.Repair
D: IgnorFatal   E: Close
------------------------------------------------------------
```

4.2.9.1 MemInfo

For historical reasons, GLEAM contains its own memory management for the elements of the chromosomes. At program start 2 MB are allocated from which the elements for the chromosomes are taken. There are two element types: List elements for chromosomes, chromosome head and genes on the one hand, and descriptor elements for segment [2, 3] management on the other hand. Elements which are no longer used are managed in free lists. If more memory than the initially allocated 2 MB is required, additional dynamic memory is requested from the operating system.

At start the display of an empty chromosome memory shows the following:

```
---------------------------------------
           Memory Statistics:

Total memory:   2048 KBytes
Maximum used:      2 KBytes   0.1 %
```

```
             List Elements:
Maximum used:     17
Present used:     17         100.0 %

          Descriptor Elements:
Maximum used:      0
Present used:      0           0.0 %

Released AKs:      0

Memory allocation:
....................................

    Legend for memory allocation:
xxx = maximum of used memory:   0.1%
### = presently used memory :   0.1%
... = unused memory          :  99.9%
--- = unavailable memory     :   0.0%
  . = 2.5% of 2048 KB = 51.2 KB

The unused element lists are ok.
No messages at state: OK
------------------------------------
```

Since the administration of the chromosome memory also works with list elements, 17 elements are used after its initialization. All further entries reflect an empty chromosome memory. Finally, the check result of the unused element lists and the error state are shown.

The memory allocation part looks different after some optimization runs:

```
------------------------------------
           Memory Statistics:

Total memory:   2048 KBytes
Maximum used:    306 KBytes  14.9 %

             List Elements:
Maximum used:   2143
Present used:   1076         50.2 %

          Descriptor Elements:
Maximum used:    303
Present used:    151         49.8 %

Released ACs:      0

Memory allocation:
###xxx.............................
```

The number of chromosomes, which are deleted from the chromosome memory is displayed in the line “Released ACs”.

4.2.9.2 Create Chr

A chromosome is created and stored in the fitness class 1 as a fitness of zero is assigned to it. This may be changed by a subsequent simulation. Note that the random number generator is not initialized at program start. This happens only if it is requested at the start of an optimization job. As long as the generator remains uninitialized, this function can be used to generate chromosomes in a reproducible way, which may be useful for certain test situations.

4.2.9.3 Check Chr

Chromosomes are manipulated by the genetic operators. If additional application oriented operators are implemented, they must be checked to produce valid chromosomes. For this purpose, this menu function is used to check the consistency of a chromosome with respect to its length and segment structure [2, 3] and the related data noted in the header:

```
-------------------------- System --------------------------
1: MemInfo      2: Create Chr   3: Check Chr    4: Chr-Editor
5: Del Chr      6: Del Chr/Cl   7: Del ChrMem
9: HamDist      A: MutOpRate    B: GenOp-Test   C: Gen.Repair
D: IgnorFatal   E: Close
------------------------------------------------------------
Select menu item: 3

              Check for Chromosome Data Consistency:

Chr [class/index]: 1
Chr [class/index]: 1/ 1

Chromosome data (length and segment structure) ok.
```

4.2.9.4 Chr-Editor

The chromosome editor is used for the creation and modification of chromosomes. For this purpose, either a new chromosome is created or one is loaded from the chromosome memory and modified. The editor is based on the nomenclature of action chains. It is not described further in the first version of this documentation.

4.2.9.5 Del Chr

This function allows the selection of a chromosome of the chromosome memory and its deletion.

4.2.9.6 Del Chr/Cl

All chromosomes of a class are erased from the memory.

4.2.9.7 Del ChrMem

The entire chromosome memory is deleted after a confirmation.

4.2.9.8 HamDist

The hamming distance of two chromosomes is calculated and displayed:

```
-------------------------- System --------------------------
1: MemInfo      2: Create Chr   3: Check Chr    4: Chr-Editor
5: Del Chr      6: Del Chr/Cl   7: Del ChrMem
9: HamDist      A: MutOpRate    B: GenOp-Test   C: Gen.Repair
D: IgnorFatal   E: Close
------------------------------------------------------------
Select menu item: 9

      Calculation of the Hamming Distance of 2 Chromosomes:

Chr [class/index]: 7
Chr [class/index]: 7/ 1
Chr [class/index]: 14
Chr [class/index]: 14/ 1

The hamming distance of the two chromosomes is: 4.735867 %
```

4.2.9.9 MutOpRate

This function is used to display the number of applications of a mutation operator [2, 3] (mutation rate) in relation to the length of a chromosome. This is demonstrated by the mutation "`par_change_ rel`" and the EVO-File "`mbf_wpar_e.evo`", see also [11]:

```
      Mutation Rate as a Function of Chromosome Length

Index of the mutation operator (0..21): 1
Number of displayed chromosome length ranges (4..30) [7]:4

               Mutation Rate of Operator 1:
Mutation rates of the 1.fitness class (0 .. 40000):
Range for chromosome length    1 ..    6:  1
                               7 ..   12:  2
                              13 ..   18:  3
                              19 ..   24:  4

Mutation rates of the 2.fitness class (40001 .. 75000):
Range for chromosome length    1 ..   11:  1
                              12 ..   18:  2
                              19 ..   26:  3
                              27 ..   33:  4

Mutation rates of the 3.fitness class (75001 .. 100000):
Range for chromosome length    1 ..   19:  1
                              20 ..   29:  2
                              30 ..   39:  3
                              40 ..   49:  4
```

For each fitness class the first four ranges of increasing numbers of mutations are displayed.

4.2.9.10 GenOp-Test

This function can be used to test genetic operators. This is particularly useful for application-specific operators that have been implemented subsequently via the standard interface of module appl_go.c from the appl package. These are addressed via negative indices, since the index range between 0 and LAST_XO is reserved for the standard operators.

First the genetic operator is selected. The range -1 to -LAST_XO was chosen for possible application-specific operator indices, since it is assumed that not more specific than general operators are implemented:

```
-------------------------- System --------------------------
1: MemInfo      2: Create Chr   3: Check Chr    4: Chr-Editor
5: Del Chr      6: Del Chr/Cl   7: Del ChrMem
9: HamDist      A: MutOpRate    B: GenOp-Test   C: Gen.Repair
D: IgnorFatal   E: Close
------------------------------------------------------------
Select menu item: b

                  Test of Genetic Operators:

Index of the genetic operator [-29..28]: 1
Chr [class/index]: 4
Chr [class/index]: 4/ 1

Fitness values: Parent: 9973.5  Child: 9772.49
Child stored at address 4/2.
```

The relative mutation operator [2, 3] was chosen and applied to a chromosome of relative poor fitness. The fitness of the offspring was slightly worsened and it is stored in the chromosome memory for further inspection.

The next example applies the standard recombination operator of GLEAM [2, 3] to two parent chromosomes of the Griewank test function. As this application works with chromosomes of fixed length the standard genetic repair is applied to the offspring. The example yields an improved first child:

```
                  Test of Genetic Operators:

Index of the genetic operator [-29..28]: 24
Chr [class/index]: 5
Chr [class/index]: 5/ 1
Chr [class/index]: 9
Chr [class/index]: 9/ 1

Parent fitness   : Parent 1: 17034.8  Parent 2: 51789.2
Offspring fitness: Child 1 : 66815.8  Child 2 : 9987.39
1st Child stored at address 10/1.
2nd Child stored at address 4/1.
```

If a non-existent application-specific operator is selected, the `appl` package detects a fatal error:

```
                   Test of Genetic Operators:

Index of the genetic operator [-29..28]: -1
Chr [class/index]: 5
Chr [class/index]: 5/ 1

Offspring was not mutated! Deleted.
 ***** 1 messages at state "FATAL":
FATAL (APPL_GO/2 ): Unknown application! Code = 0
```

For a more detailed interpretation of (fatal) error messages see Sect. 2.3.

4.2.9.11 Gen.Repair

An application dependent genetic repair is performed for a chromosome selected from the chromosome memory. At present, only the robot path planning task uses a genetic repair, see also [2, Sect. 4.5]. It is possible to implement application oriented genetic repair mechanisms via the application interface of the `appl` package, which can be used by this menu function.

4.2.9.12 IgnorFatal

A fatal error blocks some functions of the interactive program version, in particular starting an optimization job. This function resets the internal error state to allow a further investigation of the situation. Great care must be taken here, as a program crash is possible after a fatal error. In case of a previously fatal error, the following will be done:

```
--------------------------- System ---------------------------
1: MemInfo      2: Create Chr   3: Check Chr    4: Chr-Editor
5: Del Chr      6: Del Chr/Cl   7: Del ChrMem
9: HamDist      A: MutOpRate    B: GenOp-Test   C: Gen.Repair
D: IgnorFatal   E: Close
--------------------------------------------------------------
Select menu item: d

                        Reset Error State:

Error state reset. Caution! Error situation still persists.
Function "MemInfo" will set error state again!
```

The last two lines will not be displayed if the internal error state is not `GLEAM_FATAL`.

5 Log Files

Log files are used to document one or more GLEAM runs. Any errors that occur are also documented in the log files. The first example file is from the command line version. At first, the parameters of the program call are documented. This line identifies the file to be from a CLV run. A list of the loaded files follows:

```
==== GLEAM/AE V2.2.3/Off from 09/21/2020. Log file from: 22.09.2020 17:45:27 ===
Program parameter: lesak_e.exp lsk-demo-CLV -p+
Experiment file: lesak_e.exp
Initialization:
===============
  The initialization files are also searched in the GLEAM_ROOT directory: "/home/
wilfried/gleam/InitFiles"
  ActionMod  "/home/wilfried/gleam/InitFiles/mitsu_hy_e.mod" loaded.
  Evaluation "/home/wilfried/gleam/InitFiles/lesak_e.bew" loaded.
  ProgParams "/home/wilfried/gleam/InitFiles/mitsu_e.tsk" loaded.
  EvoParams  "/home/wilfried/gleam/InitFiles/lsk_stnd_e.evo" loaded.

  Kinematics-File "/home/wilfried/gleam/InitFiles/mitsu.kin" loaded.

  Obstacle-File "/home/wilfried/gleam/InitFiles/stolper.obs" loaded.
      3 obstacles successfully read.
Initialization completed.
  No chromosome file read.

---Job-- --Sizes- -------Init------- Res HamDist Rank AcceptanceRule
 No OptP Pop/Deme  RNG   Popul  Spar Chr     [%]  Par  for Offspring
  1  Evo   150/8   yes     new     0   1     0.1  1.4  localLeast-ES
-------------------------------------------------------------------
       Fitness  Runtime   Gen  GDI GAc MaxEval |
 Limit: 100000  0:05:00  2000  400 200       - |
  Init:   2162  0:00:00     0    0   0     150 |
-------------------------------------------------+---------------------------------
       Fitness  Runtime   Gen  GDI GAc MaxEval |         Job List Results
 Limit: 100000  0:05:00  2000  400 200       - | Number BestChrom.: Fitness Length
ActVal: 100000  0:00:12   555    0   0  583110 |      1              100000     27
-------------------------------------------------+---------------------------------
# 4 150 0 0  12 555 100000 27 583110

Total elapsed time: 0:12. Number of evaluations: 583110.
#t 12.033626 583110  48456.7 1050  2.0637e-05  # sec Eval Eval/sec Eval/Gen sec/Eval
Result: 1 chromosomes. Best is #1: 16/1, fitness: 100000.000000000
```

The documentation of an optimization run, called a job, lists the used job parameters at first. This is followed by the results of the initialization (see line starting with “Init:”) and the final optimization results in the line starting with “ActVal:”.

Interpretation:

- The job uses evolution only. It is based on a population size of 150 and a deme size of 8. The random generator (RNG) is initialized (yes). The initial population is generated completely randomly (new) and the strategy parameter for the initialization procedure chosen. It is set here to 0, which means that none of the generated chromosomes must have a fitness larger than zero. One result chromosome (ResChr) is required. This is followed by the values for the minimal Hamming distance (HamDist) of two parents to perform any crossover, the ranking parameter (RankPar) and the acceptance rule for offspring. Most of these parameters are explained in Sect. 2.2.

- The “Limit”-line contains the stopping criteria of this run, which can also be interpreted as limits. If one of those limits is surpassed the run is stopped. Note that there is no limit for the number of evaluations specified.

- The “Init”-line contains the results of the initialization of the 150 individuals. The resulting fitness is 2162, which is pretty low. As the robot simulator is fast the elapsed time is below one second.
- The “ActVal”-line shows the situation at the end of the job. The job was terminated, because the target fitness of 100,000, which is the maximum possible, was reached. For this 583110 offspring were evaluated in 555 generations in 12 seconds. No stagnation occurred (GDI and GAc are both zero).

The results are repeated in a more textual form below the tabular presentation. Here the final fitness is also given more precisely. Furthermore, this section contains two lines starting with a “#”-sign containing some key figures for statistical analysis, which can be extracted from a log file by appropriate tools. They will be described in the following two sections.

The logging of an adaptive MA is explained based on the last example run of Sect. 4.2.8. It also starts with a listing of the job parameters, which is extended now by the following entries:

```
---Job--  …  active UpdRate  Offspring Fitness AdaptationSpeed Learn
 No OptP  …    LHCs     [%]  Selection Classes  Param/LHC-Sel.  Rate
  1 AMMA  …      RC     100  best only       3    fast/fast    33.33
-----------------------------------------------------------------
```

This is followed by a list of all optimization parameters as for the "ShowEvoPar" menu item. After that, the job termination, initialization and ending data are shown like in simple evolution jobs. The only exception is the “`#` “ statistics line, which is extended by result data of the adaptation, see next section.

5.1 General Statistics Data Line

The general statistic line starts with “`#` “ and contains the following values for tools to perform some statistical evaluations on multiple runs contained in one log file:

data item:	**LHC job**
- acceptance rule for offspring (0=ALWAYS, 1=LOCAL_LEAST, 2=BETTER_PARENT, 3=ALWAYS_ES, 4=LOCAL_LEAST_ES)	unused
- population size	used
- last amount of generations without deme improvement (GDI)	unused 0
- last amount of generations without offspring acceptance per deme (Gac)	unused 0
- used time in seconds	used
- processed generations	unused 0
- fitness of the best chromosome	used
- length of the best chromosome	used
- number of fitness evaluations / assessed chromosomes	used

See also the software documentation of the module `evo_gsw.c` of the `evo`-package for more details.

For adaptive MA jobs this line is extended by the following data items:

- Total number of all updates
- Total number of level updates of the adaptive all-improvement (adaptive LHC optimization of all offspring)
- Highest active fitness class, followed by these data of this class:
  - Total number of updates of the levels and of the LHC probabilities, but not of adaptive all-improvement
    This is followed by the following data per LHC:
    - Probability of this LHC

- Number of the level with the highest probability for adaptive all-improvement
- Number of the level with the highest probability for all control parameter of this LHC

The MA part of the "**#** " line of the above log file is interpreted as an example:

```
# 4 20 0 0  0 8 100000 5 17915   33 0 3  29  0.9600 1 1 2  0.0400 1 2
```

A total of 33 updates of the parameter settings were made, in other words 33 adaptations were carried out. As adaptive LHC optimization of all offspring was not selected for the logged job, none have occurred. The highest fitness class was 3 and for this class most of the adaptations (29) were done. From this it can be concluded that with this application, the individuals of the relatively small population advanced rather quickly into the upper fitness area, where most evaluations and thus adaptations took place.

The data of this fitness class for the first LHC, the Rosenbrock algorithm, are a probability of 96%, and the following levels of highest probability for adaptive all-improvement, iteration limit and termination threshold (termination limit) of this LHC: 1, 1, and 2. This means that only one most used level was changed during the run in the highest fitness class, the Rosenbrock termination threshold.

Next the data of the Complex procedure show a probability of 4% and a preference of the second level for the iteration limit.

Note that both levels of highest probability for the adaptive all-improvement remained unchanged, which could not be expected otherwise,as this feature was switched off.

5.2 Performance Statistics Data Line

The performance related statistic line starts with "**#t** " and contains the following values for tools to perform some statistical evaluations on multiple runs contained in one log file:

- The precise time in seconds
- The number of evaluations performed including the initialization of the start population
- The average number of evaluations performed in one second.
- The average number of evaluations performed in one generation
- The average time required per evaluation.

As with the robot application [2, 3, 6], the time required per simulation may not be constant because it depends, for example, on the number of instructions performed and on the number of simulated cycles. Therefore, the above list refers to average values.

The statistic line contains the units of the figures after a further "#"sign:

```
#t 12.033626 583110  48456.7 1050  2.0637e-05  # sec Eval Eval/sec Eval/Gen sec/Eval
```

The above example can be read as a job duration of 12.033636 seconds, where 583,110 evaluations were performed at a rate 48,456.7 evaluations per second or 1050 evaluations per generation. One evaluation took 0.000020637 seconds.

5.3 Results of the Adaptation

In order to get a better insight into the development of the adaptation in the different fitness classes, the interactive version also includes a statistical evaluation of all jobs from the start of the program, from which histograms can easily be derived. The following data are collected for each fitness class: and LHC:

- The probability of the LHC, resulting in the "p" lines.
- The level of the highest probability of all LHC control parameters
- The level of the highest probability of adaptive all-improvement, resulting in the "A" lines.

Each histogram consists of 10 entries for the LHC probabilities and as much entries as levels were used. The data lines have the following syntax (BNF):

```
*<LHC-ID>P      <fcl> <histogram data> <jobs> <average>
*<LHC-ID><idx>P <fcl> <histogram data> <jobs> <average>
*<LHC-ID>A      <fcl> <histogram data> <jobs> <average>
```

with `<LHC-ID> = R | C` R=Rosenbrock procedure C=Complex algorithm

`<fcl>:` fitness class

`<histogram data>:` 10 integer values indicating the frequency of occurrence

`<jobs>:` number of optimization jobs involved

`<average>:` average of the histogram entries

`<idx>:` index of the adaptively controlled LHC parameter

An example of 10 AMMA runs optimizing the Griewank function[1] may illustrate this:

```
*Rp 1    0   0   0   0   2   3   2   1   1   1    10 0.659887
*R0 1   10   0   0   0   0   0   0   0   0   0    10 1
*R1 1    2   3   5   0   0   0   0   0   0        10 2.3
*RA 1   10   0   0   0   0   0                    10 1
*Cp 1    1   1   1   2   3   2   0   0   0   0    10 0.340113
*C0 1    3   4   3   0   0   0   0   0   0   0    10 2
*CA 1   10   0   0   0   0   0                    10 1

*Rp 2    0   0   0   0   1   3   3   1   1   1    10 0.669831
*R0 2    9   0   1   0   0   0   0   0   0   0    10 1.2
*R1 2    3   3   3   1   0   0   0   0   0        10 2.2
*RA 2   10   0   0   0   0   0                    10 1
*Cp 2    1   1   1   3   3   1   0   0   0   0    10 0.330169
*C0 2    4   5   0   0   0   0   0   0   0   0     9 1.55556
*CA 2    9   0   0   0   0   0                     9 1

*Rp 3    0   0   0   0   0   0   0   0   0  10    10 1
*R0 3    9   1   0   0   0   0   0   0   0   0    10 1.1
*R1 3    1   3   0   0   2   2   1   1   0        10 4.4
*RA 3   10   0   0   0   0   0                    10 1
*Cp 3   10   0   0   0   0   0   0   0   0   0    10 0
*C0 3    4   2   1   2   1   0   0   0   0   0    10 2.4
*CA 3   10   0   0   0   0   0                    10 1
```

There are three blocks of identical structure for the three fitness classes involved. Thus, only the first block is explained in detail. Each block consists of data of all involved LHCs, here two, namely the Rosenbrock procedure and the Complex algorithm. The first line shows the distribution of the final probabilities for this fitness class and the Rosenbrock algorithm. The average is about 66%. This is followed by the distribution of the most used levels of the Rosenbrock control parameters. In the given example the iteration limit level with the highest probability remained unchanged in the lowest fitness class, while the most used levels for the termination limit were changed and cover the range of levels 1 to 3 with a tendency of growth (level 3 was counted 5 times). Finally, the distribution of the 6 levels of the adaptive all-improvement is shown. The levels were left unchanged as this feature was not active in all jobs. The same is repeated for the Complex procedure, which has only one control parameter, the iteration limit.

This example shows that during the course of evolution the usage of the Complex algorithm is drastically reduced for this test function. In the end it is switched off, see the probabilities for fitness class 3. For fitness class 2 there were only 9 jobs counted for the Complex parameter and the adaptive all-improvement of this LHC. This can be caused by the fact, that during one run both values were not adapted, because e.g. the target fitness was reached before.

1: See `mbf_griew_5par_lhc.exp`

The data also shows that it may be wise to start with higher levels for the Rosenbrock termination limit. However, this cannot be concluded with certainty from the available results and would have to be verified by further runs. It could be that low values for the termination limit at the beginning of a run make more sense than higher ones and the higher ones cause too much unnecessary effort. The same applies to the question whether, given the shutdown of the Complex algorithm in the highest fitness class, it would not be better to work with the adaptive simple MA (ASMA) and the Rosenbrock method as LHC. And indeed, a simple experiment shows that 10 ASMA runs for this application require only about 2.1 million evaluations in total, compared to 6.2 million for 10 AMMA jobs. For comparison, the number of 39.2 million evaluations for 10 runs of the basic EA GLEAM should be mentioned.

6 Determination of Appropriate Population Sizes

A suitable population size μ must be determined experimentally for each application. To explain the general approach, we will draw on experience with benchmark functions whose optimum is known. Figure 6.1 shows the effort[1] (blue line), which is necessary to achieve a certain target fitness for different population sizes. With very low values of μ an otherwise high effort (dotted line) can be limited by another termination criterion. If μ is too low, more or less many runs will not reach the target fitness and are therefore considered *unsuccessful*. With rising μ the success occurs more frequently, until finally all runs are successful (*work area*). If one continues to increase μ, the effort increases likewise, without it would be necessary or useful.

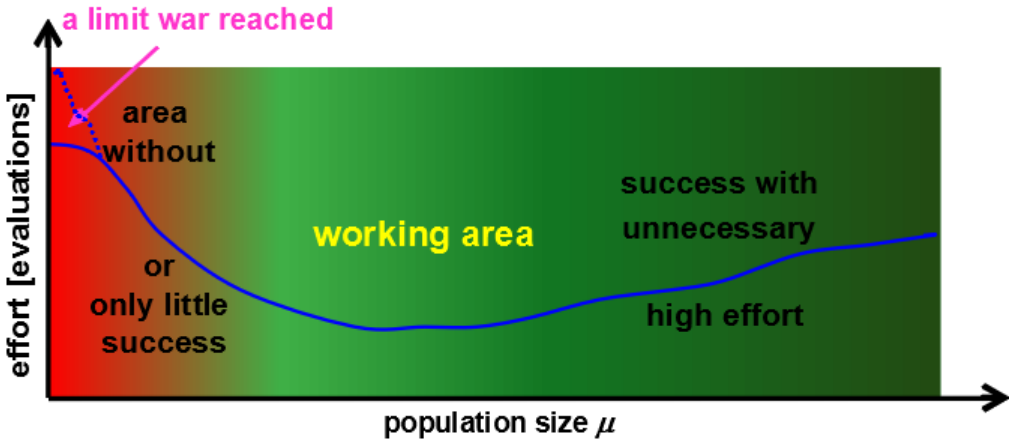

Fig. 6.1: Relationship between population size and success as well as effort

For a new application, the attainable fitness is usually unknown and it must be determined experimentally what should be considered a good solution and thus a *success*. In addition it is recommended to begin with a rather to the large μ. The determination of a suitable starting value for μ can be seen as more or less well estimated guessing. For the basic EA, values of some 100 individuals are nothing unusual, whereas for an MA, much lower values are sufficient, e.g. 1/8 or 1/10 of a good population size for the EA. The TSK-Files contain suitable population sizes for the chosen optimization procedures and some test functions, the robot path planning application and the scheduling example.

Due to the stochastic nature of EAs and MAs, several runs are required, e.g. at least 10 or better 30, if the simulation times allow it. Stagnation criteria can be used as further termination criteria, see the left dashed end of the blue curve in the area "a limit was reached".

If these runs all yield similar fitness values at similar effort, take an average value as the target fitness and the maximum of the fitness calculations plus a supplement as the effort limit. Depending on the amount of time spent, the supplement can be a small (e.g. 20%) or a larger (50-100%). Such a result also shows that the target quality for the found population size is stably reached and we are somewhere in the green zone. Larger fluctuations in effort indicate that we are close to the left transition area. If also the fitness fluctuates, we are in the red/green range and the population size is too small. It should then be increased until reliable stable results are obtained. This is then a μ in the left or middle part of the working area and thus the result we are looking for.

Otherwise try a smaller μ. If the achieved fitness remains at a similar success level and the required effort decreases without fluctuating too much, we are in the right area of the diagram of Figure 6.1 and can further reduce μ until runs occur which do not reach the success level or the effort starts to fluctuate strongly and increases on average. Thus we reached the transition to the red area and determined the lower limit for suitable values of μ.

1: The effort is measured in fitness calculations. In GLEAM this is the number of evaluated offspring, which is also contained in the column “MaxEval” in the “ActVal:” line, see also log file description an interactive run output..

7 Approximate Determination of a Pareto Front

It is well known that convex Pareto fronts can be approximated by a suitable variation of the weights of the weighted sum, see also [10]. Note that non-convex parts of the Pareto front cannot be reached by the weighted sum. As described in [10], the cascaded weighted sum can overcome this limitation to some extent.

To support automated runs with varying weights, which are started by e.g. an appropriate shell script, the `-W`-option was introduced, see also Sect. 2.1. The option can be given as often as needed for one program call to change the weights of two or more criteria. The general rule of options of one command line, which says that the last option applies, is modified here in so far as the last option regarding a certain criterion applies. E.g. `-W1:40000 -W2:55000 -W1:45000` results in 45% for criterion 1 and 55% for criterion 2.

Wrong criteria numbers will cause an error and abort the run. The same applies, when the weights of all (and not only the altered criteria!) do not sum up to the maximum fitness of 100,000.

8 Literature

[1] W. Jakob: *Applying Evolutionary Algorithms Successfully: A Guide Gained from Real-world Applications*. KIT Scientific Working Papers, Karlsruher Institut für Technologie (KIT), vol. 170, 2021, doi: 10.5445/IR/1000135763, arXiv:2107.11300
A copy is included in the literature sub directory of the documentation directory.

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

[3] 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

[4] W. Jakob: *GLEAM and HyGLEAM – Software Documentation*. Technical Paper, V5.0, IAI, 2021.
See `package-_and_SW-Docu_v5.0.pdf`

[5] W. Jakob: *Applications Included in GLEAM and Integration of New Ones.* Technical Paper, V1.0, IAI, 2021.
See `GLEAM-Applications_Docu_V1.0.pdf`

[6] W. Jakob, M. Gorges-Schleuter, C. Blume: *Application of Genetic Algorithms to Task Planning and Learning*. In: R. Männer, B. Manderick (eds.): Conf. Proc. of Parallel Problem Solving from Nature 2 (PPSN-II), Brussels, Belgium, Elsevier, Amsterdam, 1992, pp.293-302

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

[8] 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.

[9] W. Jakob: Documentation *of the Chromosome Files*. Technical Paper, V1.3, IAI, 2020.
See `Chromosomefile-Docu_V1.3.pdf`

[10] W. Jakob, C. Blume: *Pareto Optimization or Cascaded Weighted Sum: A Comparison of Concepts*. Algorithms, 7(2), 166-185, 2014. doi: 10.3390/a7010166

[11] W. Jakob: *EVO File Documentation*. Technical Paper, V1.0, IAI, 2020.
See `EVO-File-Docu_V1.0.pdf`

[12] W. Jakob: *Proofs of the Metric Axioms*. Addendum to some published papers.
Included in the literature sub directory of the documentation directory.