GAPLib Report Utility Reference

TABLE OF CONTENTS

report/--background--
report/--format--
report/DoReport
report/EndReport
report/MakeReport

PURPOSE

To enable easy generation of data files for GAP-Lib.

OVERVIEW

Report generation utility code for GAP-Lib.

The report utility package consists of three functions, one to initialize a report structure, one to write data to any report files and one to free resources allocated by the other two functions. The useage should be fairly obvious and the function and format descriptions will now follow after just one further ado.

EXAMPLE

This example outlines the useage of the report function, in this case two report files will be generated - "example.avg" and "example.max".

Example code:

```
#include <GAP.h>
#include "report.h"

int main(void)
{
struct Report *Rep;
struct Population *Pop;
...
   Rep = MakeReport("example",NULL);
   if(Rep!=NULL) {
   ...
      Pop = Evolve(Pop,Tags);
      DoReport(Rep,Pop,AVERAGE|MAX);
   ...
      EndReport(Rep);
   }
...
}
```

OVERVIEW

The data format output by DoReport() is indeed very simple. It consists of two columns of data in human-readable form. The first column is the generation and the second the data value itself. Furthermore, at the top of each file there is a one line comment describing the content of the columns.

The reason for choosing this format is one one hand that it is simple to work with, and on the other hand to be compatible with gnuplot.

DESCRIPTION

The data format is as follows (Well, actually the format for TypeCount is <Integer> <Integer>, but it does not hurt to treat it as double):

```
#Generation, <Data>
<Integer> <Double>
...
...
<Integer> <Double>
```

EXAMPLE

Sample datafile:

```
#Generation, Average
1 28.23554
2 29.45442
3 31.13943
4 31.64233
5 31.88642
6 33.19521
7 35.16777
8 35.74699
9 36.77469
10 37.45532
```

NAME

EndReport -- Free a report structure.

SYNOPSIS

```
void EndReport(struct Report *);

EndReport(Rep);
```

FUNCTION

Frees all resources associated with a report structure and flushes output. In the case of using multipass mode, no output is written until this function is called.

INPUTS

Rep - The report structure to be deallocated.

RESULT

None.

BUGS

None known.

SEE ALSO

MakeReport(), DoReport()

NAME

DoReport -- Write report data from a population structure.

SYNOPSIS

```
void DoReport(struct Report *,struct Population *,unsigned long int);

DoReport(Rep,Pop,Flags);
```

FUNCTION

Writes poulation data to a set of report files.

INPUTS

```
Rep   - The report structure in question.
Pop   - The population to retrieve the data from.
Flags - A set of flags which determine which data to write.
        Available flags are:

         AVERAGE   - Average fitness.
         MEDIAN    - Median fitness.
         TYPECOUNT - Count of most common fitness value.
         STDDEV    - Standard Deviation.
         MAX       - Maximum fitness.
         MIN       - Minimum fitness.
         ALL       - All the above.

        Flags are combined by OR:ing them together and e.g.
        AVERAGE|MAX would mean write average and maximum fitness
        to the report files.
```

RESULT

None.

BUGS

None known.

SEE ALSO

MakeReport(), EndReport()

NAME

MakeReport -- Initialize a report structure and return it.
MakeReportT -- Varargs interface to MakeReport.

SYNOPSIS

```
struct Report *MakeReport(char *,struct TagItem *);
struct Report *MakeReportT(char *,...);

Rep = MakeReport(Basename,TagList);
Rep = MakeReport(Basename,...);
```

FUNCTION

This function will initialize a report structure and open report files for writing. Basename is the first part of the filenames used for the data. The actual filenames are formed by appending a suitable suffix to the basename such as ".max" for maximum fitness data.

TAGS

REP_Multipass - (BOOL) Enables or disabled multipass mode. In multipass mode all values are stored in memory and the average over all runs are written when EndReport() is called. To envable multipass mode, one must also provide REP_Generations. Multipass mode uses substantially more memory than singlepass mode.

REP_Generations - (int) Sets the number of generations in each pass for multipass mode.

REP_Indexed - (BOOL) Usually, old reports are overwritten when running a program again. With this flag set to TRUE, every report will have an index from 0 to MAX_INDEX as defined in report.c (Currently 2048). The correct index is determined at run time and is the highest free index. An index is considered available if there is no file with the same basename and index regardless of extension.

INPUTS

Basename - The main part of the names of the report files.
TagList - Options to MakeReport().

RESULT

An initialized report structure or NULL if an error occured.

BUGS

None known.

SEE ALSO

DoReport(), EndReport()