# D.L. Carroll's FORTRAN Genetic Algorithm Driver

This is version 1.7a, last updated on 4/2/2001.
Download from: <http://cuaerospace.com/carroll/ga.html>



This genetic algorithm (GA) driver is free for public use. My only request is that the user reference and/or acknowledge the use of this driver in any papers/reports/articles which have results obtained from the use of this driver. I would also appreciate a copy of such papers/articles/reports, or at least an e-mail message with the reference so I can get a copy. Thanks.

This program is a FORTRAN version of a genetic algorithm driver. This code initializes a random sample of individuals with different parameters to be optimized using the genetic algorithm approach, i.e. evolution via survival of the fittest. The selection scheme used is tournament selection with a shuffling technique for choosing random pairs for mating. The routine includes binary coding for the individuals, jump mutation, creep mutation, and the option for single-point or uniform crossover. Niching (sharing) and an option for the number of children per pair of parents has been added. More recently, an option for the use of a micro-GA has been added.

For companies wishing to link this GA driver with an existing code, I am available for some consulting work. Regardless, I suggest altering this code as little as possible to make future updates easier to incorporate.

Any users new to the GA world are encouraged to read David Goldberg's "Genetic Algorithms in Search, Optimization and Machine Learning," Addison-Wesley, 1989.

The seven FORTRAN GA files are:
- ga170.f
- ga.inp
- ga2.inp (w/ different namelist identifier)
- ga.out
- ga.restart
- params.f
- ReadMe (this file!)

I have provided a sample subroutine "func", but ultimately the user must supply this subroutine "func" which should be your

cost function. You should be able to run the code with the sample subroutine "func" and the provided ga.inp file and obtain the optimal function value of 1.0000 at generation 187 with the uniform crossover micro-GA enabled (this is 935 function evaluations). Note that because different computers may treat precision and truncation differently, I have seen cases where two computers using the same input produce different evolution histories (but still converge to the optimal).

I still recommend using the micro-GA technique (microga=1) with uniform crossover (iunifrm=1). However, if possible, I strongly suggest that you use values of nposibl of 2**n (2, 4, 8, 16, 32, 64, etc.). While my test function works fine for other values of nposibl, I have encountered problems where the uniform crossover micro-GA has difficulty with parameters having long bit strings and a non-2**n value of nposibl, e.g. nposibl=1000, will have 10 bits assigned (for this case I would suggest running nposibl=1024 rather than 1000); I am presently investigating possible fixes for this situation.

---

Updates:

Version 1.7 includes several improvements:

(i) The coding and input files are cleaned up to provide identical output across a wider range of computers.

(ii) The arrays have been rearranged to enable a more efficient caching of system memory. For cases with very large population sizes, run time improvements of as much as a factor of 4-6 were observed! For population sizes less than 1000 you will not see much change.

(iii) A summary of the results has been added to the end of the output file.

(iv) An alternate input file "ga2.inp" has been included. Some compilers require an '&' and a '/' in the namelist input file, rather than '$' signs.

(v) For those wishing to try ever harder test functions, the included function is now N-dimensional, where N is simply determined by the number of parameters specified (nparam).

Version 1.6.5 of the code allowed creep mutations to be implemented with the micro-GA technique. (This version was never officially released.)

Version 1.6.4 of the code has a minor modification to the niching routine and another minor modification which would only affect a user having a single parameter with more than 2**30 possibilities (probably noone has used this large a number).

Version 1.6.3 of the code fixes a bug in the niching routine. Niching

should now work much better than in previous versions. A few other minor changes have been made (not worth mentioning). The sample function has been changed to something a bit more challenging.

Version 1.6.2 of the code has had major restructuring in the form of converting all of the operators (crossover, mutation, etc.) into subroutines. The code logic should be a little more understandable now and it lends itself to more easily modifying parts of the code. The counter kountmx (see v1.6.1 comments below) was added to the namelist input. Otherwise, code performance should be the same.

Version 1.6.1 of the code has very minor modifications. If you are already successfully using the code, then you will not need this update.

(i) Added a little documentation about changing format statements 1050, 1075, 1275, and 1500 when you change nparam or the total number of chromosomes (see below).

(ii) I have commented out all of the lines of code dealing with cputime. The Macintosh specific SECNDS call was causing more questions than I had anticipated. However, other than commenting the lines out, I have left them in their location for reference in case the user wants a cputime added.

(iii) I have included a sample output file.

(iv) Added counter (kountmx) to control how frequently the restart file is written. This saves I/O time and wear and tear on storage device. Presently set to write every fifth generation.

Version 1.6 of the code has incorporated the ability to use a micro-GA approach; this significantly reduced the number of function evaluations to find the global maximum of my test function.

Version 1.5 of the code has added some more flexibility to your available options:

(i) You now specify the minimum and maximum values of the parameters rather than the minimum and the increment.

(ii) You now specify the number of possibilities you want for each parameter, not the number of bits. This modification has two features: first, the program automatically calculates the number of bits per parameter; second, you are no longer forced to have a number of possibilities equal to 2**n. While the code is more efficient when there 2**n possibilities per parameter, it will run quite well with a lesser number; e.g. a colleague has 25 specific airfoil families he wants to investigate, greater than 16, less than 32.

(iii) You can now specify specific parameters for niching. Earlier versions of the code forced you to niche on all parameters. Now, the input array 'nichflg' permits you to choose the parameters for niching.

(iv) You have an input flag to prevent the printing of specific jump. and creep mutation information
(v) You now specify the maximum values of population size, number of parameters and number of chromosomes in an include file (params.f). This sets the maximum array sizes in the code. When running, the code only uses the array size up to npopsiz and nparam (from ga.inp) and nchrome (computed internally from the nposibl input array).

---

The code is presently set for a maximum population size of 200, 30 chromosomes (binary bits) and 2 parameters. These values can be changed in params.f as appropriate for your problem. Correspondingly you will have to change a few 'write' and 'format' statements if you change nchrmax and/or nparmax. In particular, if you change nchrome and/or nparam, then you should change the 'format' statement numbers 1050, 1075, 1275, and 1500. For example, if you have a problem with 4 parameters and 16 chromosomes (bits), then you should change these format statements to be:

```
 1050 format(1x,' #      Binary Code',8x,'Param1 Param2 Param3',
     +          ' Param4 Fitness')
 1075 format(i3,1x,16i1,4(1x,f6.2),1x,f6.2)
 1275 format(/' Average Values:',10x,4(1x,f6.2),1x,f6.2/)
 1500 format(i5,3x,16i2)
```

The CPU time related lines of code reference a Macintosh specific time function (SECNDS). To avoid compiler errors with other computers, I have commented out these lines of code. If you wish to have cputime output,then you will have to change the time functions for the specific computer you are running on. Most modern Unix machines will recognize the 'etime' function; these lines are added to the code along with the variable 'tarray' and 'cpu...again, to avoid compiler errors with different computers, these lines of code are also commented out.

A common problem arises with the Microsoft PowerStation compiler, i.e., PowerStation does not recognize the abbreviation NML for NAMELIST. If you are using PowerStation, you will likely have to substitute NAMELIST for all instances of NML.

Please feel free to contact me with questions, comments, or errors (hopefully none of latter).

Enjoy!

David L. Carroll
CU Aerospace
2004 South Wright Street Extended
Urbana, IL 61802

e-mail: carroll@cuaerospace.com
Phone:  217-333-8274
fax:    217-244-7757

##########################################################################

micro-GA Tip:

My favorite GA technique is still the micro-GA.  At this point, I recommend using the micro-GA with uniform crossover and a small population size.  The following inputs gave me excellent performance:

```
microga = 1
npopsiz = 5
maxgen  = 100
iunifrm = 1
```

I have also gotten good performance with the single-point crossover (iunifrm=0),
micro-GA.

If you decide to use the micro-GA, you will not need to worry about the population sizing or creep mutation tips below.

See the Krishnakumar reference below for more information about micro-GA's.

##########################################################################

Population Sizing Tip:

I've had a lot of people ask me about population sizing, especially people who are attempting large problems where 100 individuals is probably not enough.  The true authority on the subject is David Goldberg, but here is a crude population scaling law in my paper (based on Goldberg & Deb, 1992):

```
npopsiz = order[(l/k)(2**k)] for binary coding
```

where l = nchrome and k is the average size of the schema of interest (effectively the average number of bits per parameter, i.e. approximately equal to nchrome/nparam, rounded to the nearest integer).  I find that when I have uniform crossover and niching turned on (which I recommend doing), that this scaling law is usually overkill, i.e. you can most likely get by with populations at least twice as small.

Remember to make the parameter 'indmax' (in 'params.f') greater than or equal to 'npopsiz'.

##########################################################################

Creep Mutation Probability Tip:

I generally like to have approximately the same number of creep mutations and jump mutations per generation. Using basic probabilistic arguments, it can be shown that you will get approximately the same number of creep and jump mutations when

pcreep = (nchrome/nparam) * pmutate

where pmutate (the jump mutation probability) is 1/npopsiz.

##########################################################################

Suggested reading that I have found to be of use:

Goldberg, D. E., and Richardson, J., "Genetic Algorithms with Sharing for Multimodal Function Optimization," Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms, 1987, pp. 41-49.

Goldberg, D. E., "Genetic Algorithms in Search, Optimization and Machine Learning," Addison-Wesley, 1989.

Goldberg, D. E., "A Note on Boltzmann Tournament Selection for Genetic Algorithms and Population-Oriented Simulated Annealing," in: Complex Systems, Vol. 4, Complex Systems Publications, Inc., 1990, pp. 445-460.

Goldberg, D. E., "Real-coded Genetic Algorithms, Virtual Alphabets, and Blocking," in: Complex Systems, Vol. 5, Complex Systems Publications, Inc., 1991, pp. 139-167.

Goldberg, D. E., and Deb, K., "A Comparitive Analysis of Selection Schemes Used in Genetic Algorithms," in: Foundations of Genetic Algorithms, ed. by Rawlins, G.J.E., Morgan Kaufmann Publishers, San Mateo, CA, pp. 69-93, 1991.

Goldberg, D. E., Deb, K., and Clark, J. H., "Genetic Algorithms, Noise, and the Sizing of Populations," in: Complex Systems, Vol. 6, Complex Systems Pub., Inc., 1992, pp. 333-362.

Krishnakumar, K., "Micro-Genetic Algorithms for Stationary and Non-Stationary Function Optimization," SPIE: Intelligent Control and Adaptive Systems, Vol. 1196, Philadelphia, PA, 1989.

Syswerda, G., "Uniform Crossover in Genetic Algorithms," in: Proceedings of the Third International Conference on Genetic Algorithms, Schaffer, J. (Ed.), Morgan Kaufmann Publishers, Los Altos, CA, pp. 2-9,

1989.

##########################################################################

If you are interested in my work (which may give some insights into how and why I coded some aspects of my GA), I can mail copies of three papers of mine.

G. Yang, L.E. Reinstein, S. Pai, Z. Xu, and D.L. Carroll, "A new genetic algorithm technique in optimization of permanent 125-I prostate implants," Medical Physics, Vol. 25, No. 12, 1998, pp. 2308-2315.

Carroll, D. L., "Chemical Laser Modeling with Genetic Algorithms," AIAA J., Vol. 34,  2, 1996, pp.338-346.
(A preprint version of this paper can now be downloaded in PDF format via my website:
<http://cuaerospace.com/carroll/gatips.html> look for AIAA1996.pdf)

Carroll, D. L., "Genetic Algorithms and Optimizing Chemical Oxygen-Iodine Lasers," Developments in Theoretical and Applied Mechanics, Vol. XVIII, eds. H.B. Wilson, R.C. Batra, C.W. Bert, A.M.J. Davis, R.A. Schapery, D.S. Stewart, and F.F. Swinson, School of Engineering, The University of Alabama, 1996, pp.411-424.
(This paper can now be downloaded in PDF format via my website:
<http://cuaerospace.com/carroll/gatips.html> look for SECTAM18.pdf)

##########################################################################

Disclaimer:  this program is not guaranteed to be free of error (although it is believed to be free of error), therefore it should not be relied on for solving problems where an error could result in injury or loss.  If this code is used for such solutions, it is entirely at the user's risk and the author disclaims all liability.