LA-UR-98-3578

Title:

GENES 2.0 USER'S GUIDE: A GUIDE TO A FUZZY LOGIC BASED MULTIOBJECTIVE GENETIC ALGORITHM IN FORTRAN 77

Brian J. Reardon

Submitted to:

http://lib-www.lanl.gov/la-pubs/00412972.pdf

Los Alamos
NATIONAL LABORATORY

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the University of California for the U.S. Department of Energy under contract W-7405-ENG-36. By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. The Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; therefore, the Laboratory as an institution does not endorse the viewpoint of a publication or guarantee its technical correctness.

Form No. 836 R5
ST 2629 10/91

GENES 2.0 USER'S GUIDE: A GUIDE TO A FUZZY LOGIC BASED MULTIOBJECTIVE GENETIC ALGORITHM IN FORTRAN 77

Brian J. Reardon, Los Alamos National Laboratory, MST-6, Los Alamos, NM 87545

Abstract

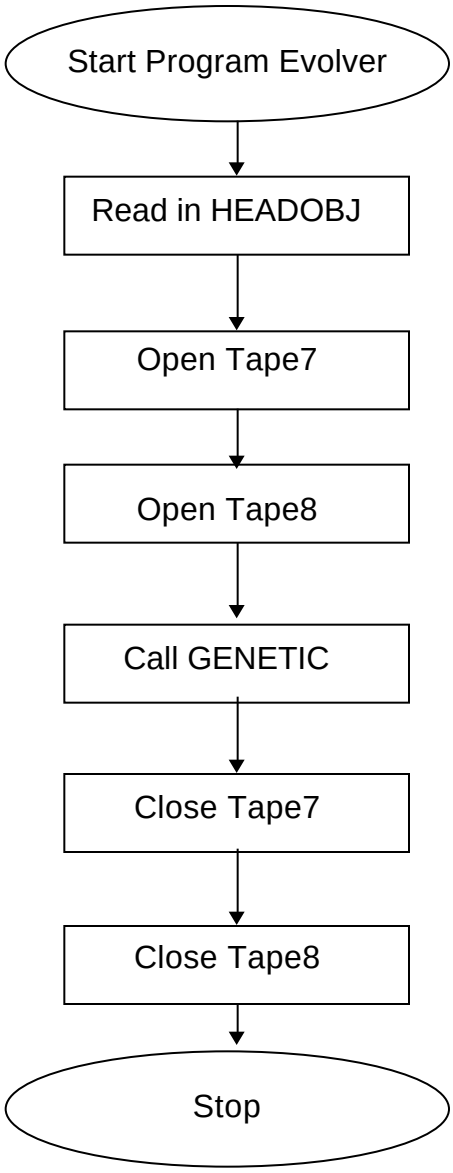
The FORTRAN 77 code listing for a fuzzy logic based multiobjective genetic algorithm is documented along with the necessary input files and descriptions of the output files.

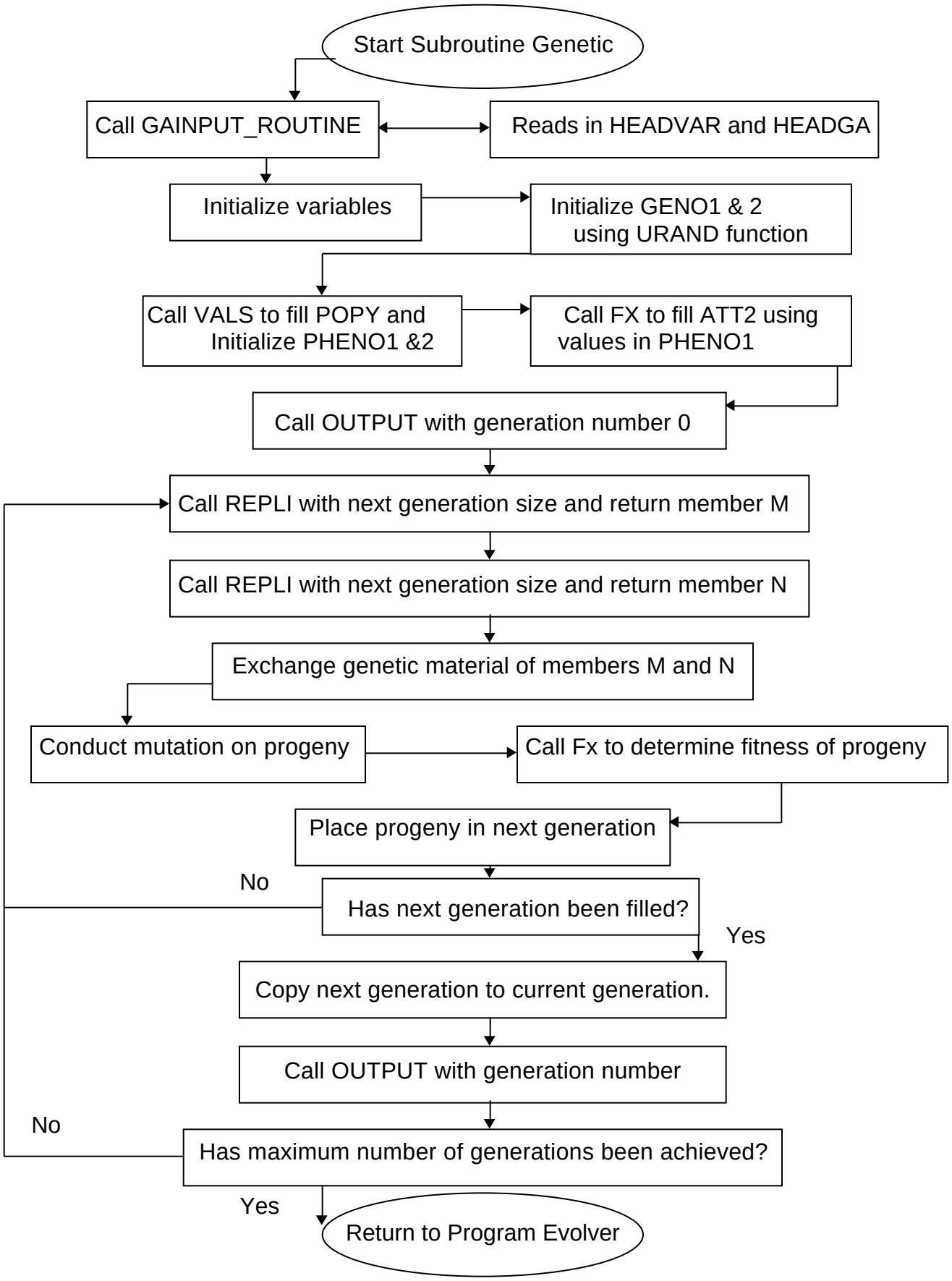
1.0 Introduction

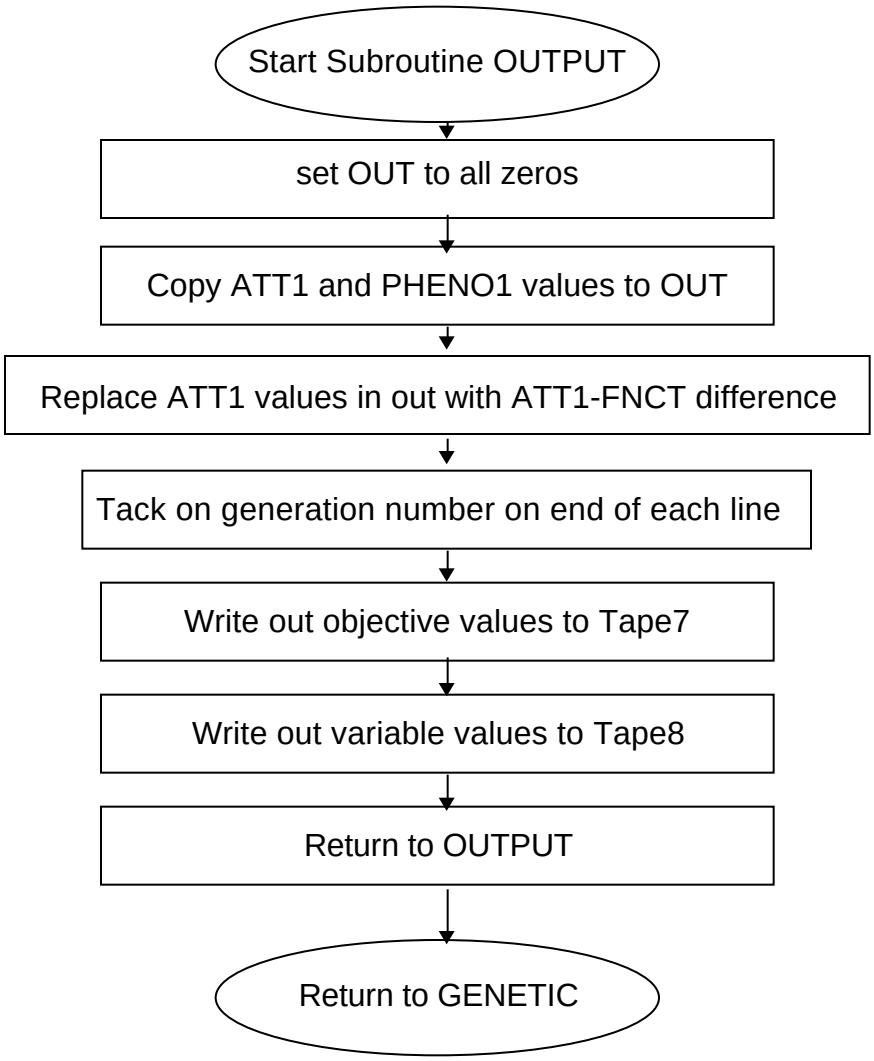
This report documents the FORTRAN 77 code listing for a fuzzy logic based multiobjective genetic algorithm that has been used previously and will be used in the future for more diverse optimization projects. The details of a genetic algorithm and the idiosyncrasies of a fuzzy logic based multiobjective selection method have been outlined in the references provided in section 3. Section 2.1 provides a general flow chart of program. Sections 2.2-4 describe the functional details of the program on a file by file and subroutine by subroutine basis. Appendix A lists the code that can be compiled as is on most platforms with a FORTRAN 77 compiler.

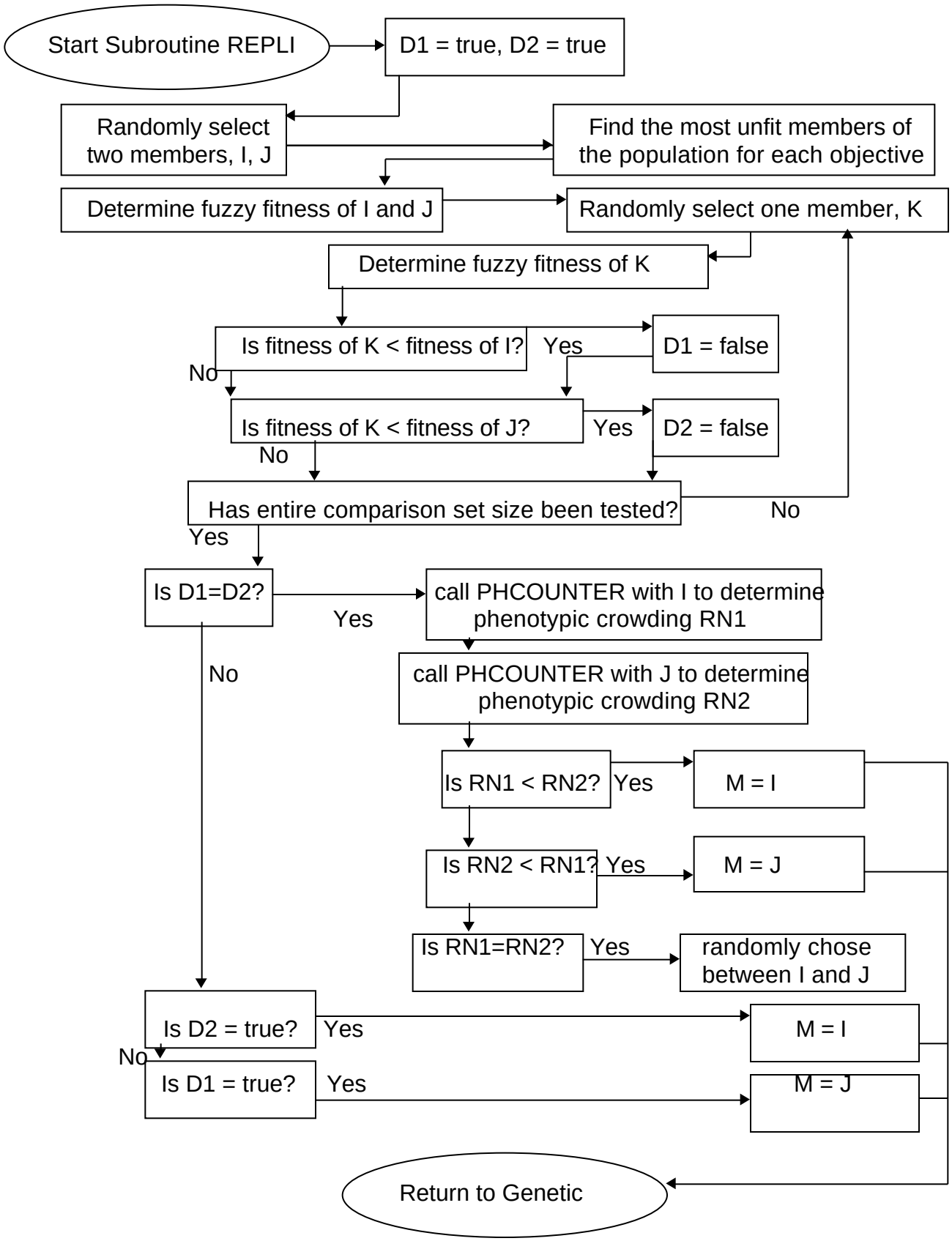
2.0.0 Program Description

2.1.0 Flow Chart




Start Subroutine Genetic
Call GAINPUT_ROUTINE
Reads in HEADVAR and HEADGA
Initialize variables
Initialize GENO1 & 2 using URAND function
Call VALS to fill POPY and Initialize PHENO1 &2
Call FX to fill ATT2 using values in PHENO1
Call OUTPUT with generation number 0
Call REPLI with next generation size and return member M
Call REPLI with next generation size and return member N
Exchange genetic material of members M and N
Conduct mutation on progeny
Call Fx to determine fitness of progeny
Place progeny in next generation
No
Has next generation been filled?
Yes
Copy next generation to current generation.
Call OUTPUT with generation number
No
Has maximum number of generations been achieved?
Yes
Return to Program Evolver


Start Subroutine OUTPUT
set OUT to all zeros
Copy ATT1 and PHENO1 values to OUT
Replace ATT1 values in out with ATT1-FNCT difference
Tack on generation number on end of each line
Write out objective values to Tape7
Write out variable values to Tape8
Return to OUTPUT
Return to GENETIC


Start Subroutine REPLI
D1 = true, D2 = true
Randomly select two members, I, J
Find the most unfit members of the population for each objective
Determine fuzzy fitness of I and J
Randomly select one member, K
Determine fuzzy fitness of K
Is fitness of K < fitness of I?
Yes
D1 = false
No
Is fitness of K < fitness of J?
Yes
D2 = false
No
Has entire comparison set size been tested?
No
Yes
Is D1=D2?
Yes
call PHCOUNTER with I to determine phenotypic crowding RN1
No
call PHCOUNTER with J to determine phenotypic crowding RN2
Is RN1 < RN2?
Yes
M = I
Is RN2 < RN1?
Yes
M = J
Is RN1=RN2?
Yes
randomly chose between I and J
Is D2 = true?
Yes
M = I
No
Is D1 = true?
Yes
M = J
Return to Genetic

2.2.0 Input

2.2.1 Input File: HEADOBJ

The input file HEADOBJ lists all the data needed by subroutine FX to calculate the objective values. The format of this file is specified by the user and is read in the main program, EVOLVER.

2.2.2 Input File: HEADGA

The input file HEADGA is read by the subroutine GAINPUT_ROUTINE and has the format:

`14`	The Length of the binary string for each variable (must be even)
`100`	The number of generations
`2`	The population size (must be even)
`1000`	The mutation rate. Here, probability of mutation is 1 in 1000.
`1`	The comparison set size.

2.2.3 Input File: HEADVAR

The input file HEADVAR lists the number of variables being optimized first and then lists the upper and lower search ranges for each variable. The HEADVAR file for a micromechanical densification problem is:

`19`	Number of variables
`2.50 3.00`	surface energy (E*omega^.33)/25 +/- 30%
`130.0 140.0`	yield stress (MPa) modulus/1000
`0.40 0.60`	t dependance of yield
`3.00 6.00`	power law creep exponent
`50.0 80.0`	reference stress P-L Creep MPa (yield stress/2)
`450.0 550.0`	act enrgy P-L creep same as volume diffusion
`1600.0 1700.0`	low temp to high temp creep transition (Tm/2)
`0.60 0.70`	c for LT creep
`0.50e-4 1.00e-2`	pre VD
`400.0 700.0`	act VD
`2.00e-14 1.00e-13`	pre BD
`50.0 500.0`	act BD
`7.00e-10 8.00e-10`	pre SD
`400.0 500.0`	act SD
`1.00e-14 2.00e-14`	pre BM
`300.0 400.0`	act BM
`7.0e-6 11.0e-6`	radius
`0.70 0.90`	stage 2 cut off density
`0.85 0.99`	stage 1 cut off density

2.3.0 Output

2.3.1 Output file: Tape7

Output file TAPE7 lists the objective values for every member of the population for each generation. Generation number is indicated by the last number in the row. The objectives listed are the real differences between the experimentally observed values and the calculated values.

2.3.2 Output file: Tape8

Output file TAPE8 lists the variable values for every member of the population for each generation. As in TAPE7, generation number is indicated by the last number in the row.

2.4.0 Programs, Subroutines and Functions

2.4.1 Program EVOLVER

Description:
EVOLVER is the main program of GENES 2.0 and is responsible for reading in the variables used in the objective functions of the GA. Furthermore, it is here that the main output files Tape7 and Tape8 are opened and closed.

Includes:
PARAMSGA
PARAMSOBJ
COMMONSOBJ
COMMONSGA

Files Opened:
HEADOBJ
Tape7
Tape8

Files Closed:
HEADOBJ
Tape7
Tape8

Subroutine calls:
GENETIC

2.4.2 Subroutine GENETIC

Description:

Genetic is the main subroutine of the genetic algorithm. This subroutine tests to make sure that the user input parameters are within the allocated limits and then initializes the Generation 0. Once generation 0 is created, the evolution loop starts. This loop calls REPLI to determine which members are most fit and then performs crossover and mutation until the next generation is filled.

Called by:
EVOLVER

Usage:
GENETIC

Includes:
PARAMSGA
PARAMSOBJ
COMMONSOBJ
COMMONSGA

Function Calls
URAND(ISEED)

Subroutine Calls
GAINPUT_ROUTINE
VALS
OUTPUT
REPLI
FX
PHCOUNTER

2.4.3 Subroutine OUTPUT

Description:
The output subroutine copies the objective and variable data for that particular generation to the OUT array and then calls OUTFILE to write the data out to TAPE 7 and TAPE 8.

Called by:
GENETIC

Usage:
OUTPUT(J)
J - the current Generation number

Includes:
PARAMSGA
PARAMSOBJ
COMMONSOBJ

COMMONSGA

Subroutine Calls
OUTFILE

2.4.4 Subroutine VALS

Description:
Subroutine VALS determines the real value of binary encoded string for member J and stores the values in POPY

Called by:
GENETIC

Usage:
VALS(J)
J - a member of the population

Includes:
PARAMSGA
PARAMSOBJ
COMMONSOBJ
COMMONSGA

Function Calls
DECIMAL

2.4.5 Subroutine OUTFILE

Description:
Determines the difference between the calculated and experimental values and then outputs the objective values to TAPE7 and the variables to TAPE8.

Called by:
OUTPUT

Usage:
OUTFILE(J)
J - generation number

Includes:
PARAMSGA
PARAMSOBJ
COMMONSOBJ
COMMONSGA

2.4.6 Subroutine GAINPUT_ROUTINE

Description:
Subroutine GAINPUT_ROUTINE reads in the variable search ranges from HEADVAR and the user specified GA parameters from HEADGA.

Called by:
GENETIC

Usage:
GAINPUT_ROUTINE

Includes:
PARAMSGA
PARAMSOBJ
COMMONSOBJ
COMMONSGA

Files Opened:
HEADVAR
HEADGA

Files Closed:
HEADVAR
HEADGA

2.4.7 Subroutine REPLI

Description:
The subroutine REPLI selects a member of the population for crossover using fuzzy logic and/or niching.

Called by:
GENETIC

Usage:
REPLI (MAX,MEM)
MAX - number of members in the partially filled next generation + 1
MEM - member of the current generation selected for crossover

Includes:
PARAMSGA
PARAMSOBJ
COMMONSOBJ
COMMONSGA

Function Calls

URAND
AVG
PHCOUNTER

2.4.8 Subroutine FX

Description:
Subroutine FX can contain the actual objective functions being evaluated or a call to the necessary subroutines. What is important is that the objective values are assigned the to the array ATT2(MEM,J) where MEM is the member of the population being evaluated and J is a particular function or objective.

Called by:
GENETIC

Usage:
FX(MEM)
MEM - member of the population whose fitness is being evaluated

Includes:
PARAMSGA
PARAMSOBJ
COMMONSOBJ
COMMONSGA

Function Calls:
User specified

Subroutine Calls:
User specified

Files Opened:
User Specified

Files Closed:
User Specified

2.4.9 Function URAND

Description:
URAND is a function written by John Kieffer at the University of Illinois Urbana-Champaign to create a pseudo-random list of random numbers between 0 and 1.

Called by:
GENETIC
REPLI

Usage:
URAND(J)
J - a large odd integer, usually the time in seconds on the system clock.

2.4.10 Function DECIMAL

Description:
Function DECIMAL returns the decimal equivalent of a binary string.

Called by:
VALS

Usage:
DECIMAL (K,J)
K - variable identification number
J - member of the population

Includes:
PARAMSGA
PARAMSOBJ
COMMONSOBJ
COMMONSGA

2.4.11 Function PHCOUNTER

Description:
Function PHCOUNTER returns a value for how crowded a member (N) is if placed in the partially filled next generation of size MAX. This is done using a normalized squares difference distance. If the distance falls within a value specified by the SCALE parameter, then the distance is counted in the crowding factor.

Called by:
REPLI

Usage:
PHCOUNTER(N,MAX)
N - The member of the current population being considered in Niching
MAX - the number of members currently in the partially filled next generation.

Includes:
PARAMSGA
PARAMSOBJ
COMMONSOBJ
COMMONSGA

2.4.12 Function AVG

Description:
The function AVG is the heart of the fuzzy logic selection method. The coordinates listed in the function call define the five major points in a fuzzy logic rule set:

```
1\                 /5
   \              /
    \            /
     2__3__4
```

AVG determines between which two points the value of X lies and then calls POINT to determine the corresponding value of Y. AVG also checks to make sure that none of the 5 points are out of their relative acceptable ranges.

Called by:
REPLI

Usage:
AVG(X1,Y1,X2,Y2,X3,Y3,X4,Y4,X5,Y5,X)
X1 - 1st point x coordinate
Y1 - 1st point y coordinate
X2 - 2nd point x coordinate
Y2 - 2nd point y coordinate
X3 - 3rd point x coordinate
Y3 - 3rd point y coordinate
X4 - 4th point x coordinate
Y4 - 4th point y coordinate
X5 - 5th point x coordinate
Y5 - 5th point y coordinate
X - The x coordinate that AVG sends to POINT to determine the Y coordinate

Function Calls
POINT

2.4.13 Function POINT

Description:
The function POINT sends back a Y value given two points on a line and a value of X.

Called by:
AVG

Usage:
POINT(X1,Y1,X2,Y2,X)
X1 - 1st point x coordinate

Y1 - 1st point y coordinate
X2 - 2nd point x coordinate
Y2 - 2nd point y coordinate
X - for this value of X and the two give points, a value of Y is determined.

2.4.14 Included File PARAMSGA

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cc Parameter File for GA routines.
cc MXNV - maximum number of variables to be optimized
cc MXNFNCT - maximum number of objectives.
cc MXLEN - Maximum binary string length
cc MXPOP - maximum populations size
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
PARAMETER(MXNV=19)
PARAMETER(MXNFNCT=20)
PARAMETER(MXLEN=14)
PARAMETER(MXPOP=400)
```

2.4.15 Included File PARAMSOBJ

If the objective subrountines have global parameters that must be specified it is often easiest to specify them here. The PARAMSOBJ file must be present when compiling the program but depending on the objective problems being addressed, it may be empty.

2.4.16 Included File COMMONSGA

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cc  The COMMONSGA file contains all the common blocks used by the
cc  GA
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cc  GAINPUT  contains the input that the GA needs to work with.
cc    IPOPU   population size
cc    MUTATE  1/MUTATE is the mutation probability
cc    NV      number of variables
cc    LEN     string length per variable
cc    IGEN    number of generations
cc    RMAXMIN(2,MXNV) Max and Min search range of variables
cc    ISEED   random integer used as seed in URAND
cc    NFNCT   number of objectives
cc    FNCT(MXNFNCT)   array of objective values
cc    ERROR(MXNFNCT)  array of uncertainty of obj. values
cc    DIFFER(MXNV)    array of differences between Max and Min search range
cc    MAXLEN(MXLEN+1) array of the integral powers of 2
cc    ICOMP           comparison set size
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      COMMON /GAINPUT/ IPOPU,MUTATE,NV,LEN,IGEN,RMAXMIN(2,MXNV),
     & ISEED,NFNCT,FNCT(MXNFNCT),ERROR(MXNFNCT),
     & DIFFER(MXNV),MAXLEN(MXLEN+1),ICOMP

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cc GAOUTPUT  the variables needed to out put data
cc  OUT array that lists the variable and function values
cc      for all members of the population for that generation
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      COMMON /GAOUPUT/ OUT(MXPOP,MXNV+MXNFNCT+1)

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cc GENGEN  common block of internal parameters and arrays used by
cc  the GA that the user can not adjust.
cc  GENO1(MXPOP,MXNV*MXLEN), genotypes of generation x
cc  GENO2(MXPOP,MXNV*MXLEN), genotypes generation X+1
cc  TGENO1(2,MXNV*MXLEN), temporary array for crossover
cc  PHENO2(MXPOP,MXNV),      phenotyes of generation x
cc  PHENO1(MXPOP,MXNV), phenotypes of generation X+1
cc  ATT1(MXPOP,MXNFNCT), attributes of generation X
```

```
      cc  ATT2(MXPOP,MXNFNCT), attributes of generation X+1
      cc  POPY(MXNV)array of variables being used in current FX evaluation
      cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

            COMMON /GENGEN/
           & GENO1(MXPOP,MXNV*MXLEN),
           & GENO2(MXPOP,MXNV*MXLEN),
           & TGENO1(2,MXNV*MXLEN),
           & PHENO2(MXPOP,MXNV),
           & PHENO1(MXPOP,MXNV),
           & ATT1(MXPOP,MXNFNCT),
           & ATT2(MXPOP,MXNFNCT),
           & POPY(MXNV)
```

2.4.17 Included File: COMMONSOBJ

Like the PARAMSOBJ file, this file must be present at compilation but only needs to contain common blocks if the objective subroutines require them.

3.0 References

Goldberg, D. E., “Genetic Algorithms in Search, Optimization, & Machine Learning,” Addison-Wesley Co., Inc., 1989.

Reardon, B. J., "Fuzzy Logic Vs. Niched Pareto Multiobjective Genetic Algorithm Optimization: Part I. Schaffer's F2 Problem," LA-UR-97-3675, Los Alamos National Laboratory, Los Alamos, NM 87544. September 1997.

Reardon, B. J., "Fuzzy Logic Vs. Niched Pareto Multiobjective Genetic Algorithm Optimization: Part II. A Simplified Born-Mayer Problem," LA-UR-97-3676, Los Alamos National Laboratory, Los Alamos, NM 87544. September 1997.

4.0 Acknowledgments

Funded by the U.S. Department of Energy and Los Alamos National Laboratory which is operated by the University of California under contract number W-7405-ENG-36.

Appendix A: Code Listing

```
************************************************************************
************************************************************************
      PROGRAM EVOLVER
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cc  This is the main program which reads in the objective data from
cc    HEADOBJ.
cc  TAPE7 and TAPE8 are opened so that the GA can write the objective
cc    values and optimized paramters values while executing
cc  PARAMSGA contains the user definded parameters for the GA
cc  PARAMSOBJ contains the user definded paramters for the Objectives
cc  COMMONSGA contains all the common blocks used by the GA
cc  COMMONSOBJ contains all the common blocks used by the objectives
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      INCLUDE 'PARAMSGA'
      INCLUDE 'PARAMSOBJ'
      INCLUDE 'COMMONSOBJ'
      INCLUDE 'COMMONSGA'
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C     READING THE HEADOBJ FILE
       OPEN (UNIT=10,FILE='HEADOBJ',STATUS='OLD')
       READ(10,*) NFNCT
       DO 259 K = 1,NFNCT
```

```
                READ(10,*) TIMER(K),FNCT(K),ERROR(K)
                DO J = 1,INT(TIMER(K))
                READ(10,*) PROCESS(K,J,1),PROCESS(K,J,2),PROCESS(K,J,3)
                END DO
 259    CONTINUE
        READ(10,*) RGCTHEORDEN
        READ(10,*) RGCTM
        READ(10,*) RGCATMW
       READ(10,*) RGCWATMV
       READ(10,*) RGCR
       READ(10,*) RGCRRATIO
        READ(10,*) RGCGDIA
       READ(10,*) RGCDENO
        CLOSE(10)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C    CALLING THE GENETIC ALGORITHM
       OPEN (UNIT=30,FILE='TAPE7',STATUS='OLD')
       OPEN (UNIT=40,FILE='TAPE8',STATUS='OLD')
       CALL GENETIC
       CLOSE (UNIT=40)
       CLOSE (UNIT=30)
       END
************************************************************************
************************************************************************
       SUBROUTINE FX(MEM)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cc The subrountine FX is called by the GA every time the objectives need
cc to be evaluated.  The variable MEM is the particular member of the
cc population whose objectives are being optimized.
cc The array ATT2(population size, number of objectives) needs to be
cc assigned the calculated objective value.  The subroutine fuzzy()
cc compares the calcuated value to the experimental value
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        INCLUDE 'PARAMSOBJ'
       INCLUDE 'PARAMSGA'
        INCLUDE 'COMMONSOBJ'
       INCLUDE 'COMMONSGA'
Ccccccccccccccccccccccccccccccccccccccc
cc the real variable values of the POPY() array are assigned to
cc the variables used in the objective calculations.
        RGCSURNRG=    POPY(1)
       RGCSY=         POPY(2)
       RGCYT=         POPY(3)
       RGCPCLEXP=     POPY(4)
       RGCPCLREFS=    POPY(5)
       RGCPCLACTNRG= POPY(6)
       RGCLT2HT=      POPY(7)
        RGCLTCC=      POPY(8)
       RGCPREVD=      POPY(9)
       RGCACTVD=      POPY(10)
       RGCPREBD=      POPY(11)
       RGCACTBD=      POPY(12)
        RGCPRESD=       POPY(13)
        RGCACTSD=       POPY(14)
        RGCPREBM=       POPY(15)
        RGCACTBM=       POPY(16)
       RGCR=          POPY(17)
       CUT1 =         POPY(18)
       CUT2 =         POPY(19)
cccccccccccccccccccccccccccccccccccccccccccccccc
cc each of the objectives are then caculated and
cc inserted into the ATT2() array.  The integer
cc value IFUNCT is carried over to the objective subrountine
cc by a common block to tell the routine which objective to
cc calculate.
        DO 10 J = 1,NFNCT
              IFUNCT = J
              ATT2(MEM,J)=HIP()
 10    CONTINUE
       RETURN
       END
************************************************************************
************************************************************************
      SUBROUTINE GENETIC
************************************************************************
************************************************************************
        INCLUDE 'PARAMSGA'
        INCLUDE 'PARAMSOBJ'
        INCLUDE 'COMMONSOBJ'
        INCLUDE 'COMMONSGA'
C
```

```
       INTEGER time
C
       CALL GAINPUT_ROUTINE
C
ccc Test to make sure that the user specified GA parameter are
ccc within the allocated limits
      IF (
     &(NV.GT.MXNV).OR.
     &(NFNCT.GT.MXNFNCT).OR.
     &(LEN.GT.MXLEN).OR.
     &(IPOPU.GT.MXPOP)) THEN
       PRINT*,'ERROR'
          STOP
      ENDIF

ccc  MAXLEN is an array of integral powers of two.
ccc  used to covert the binary sequences into real numbers
      DO 84 J = 0,LEN
       MAXLEN(J+1) = 2**J
 84   CONTINUE

ccc rmaxmin is the conversion factor for changeing between the
ccc decimal equivalent binary value and the real value.
      DO 23 J = 1,NV
       DIFFER(J) = (RMAXMIN(1,J) - RMAXMIN(2,J))
       RMAXMIN(1,J) = DIFFER(J)/(MAXLEN(LEN+1)-1)
 23   CONTINUE

Ccc INITIALIZE RANDOM NUMBER GENORATOR
      ISEED = time()
      ISEED = ABS(ISEED)
      IF (MOD(ISEED,2).EQ.0) ISEED = ISEED+1

Ccc INITIALIZE INITIAL POPPULATION
      DO 20 I = 1, IPOPU
       DO 20 J = 1,NV*LEN
        GENO1(I,J) = ANINT(URAND(ISEED))
        GENO2(I,J) = GENO1(I,J)
 20    CONTINUE

ccc Using initialize GENO array, call VALS to determine the POPY
ccc array and then assign POPY values to PHENO
       DO 41 J = 1,IPOPU
       CALL VALS(J)
       DO 41 I = 1,NV
        PHENO1(J,I) = POPY(I)
        PHENO2(J,I) = PHENO1(J,I)
 41    CONTINUE

Ccc  Given the initial variable values in PHENO, determine the objective
ccc  values for each member of the populations
      DO 22 J=1,IPOPU
        DO 27 I=1,NV
          POPY(I) = PHENO1(J,I)
 27     CONTINUE
        CALL FX(J)
        DO 210 I = 1,NFNCT
          ATT1(J,I) = ATT2(J,I)
 210    CONTINUE
 22   CONTINUE

Ccc write out the results of the initial generation, 0
       CALL OUTPUT(0)

Ccc START GENETICS

      DO 10 J = 1, IGEN
       PRINT*,'GENERATION #',J

C   SELECTING MEMBERS TO BE CROSSED
       DO 97 I = 1,IPOPU,2
ccc         M and N are returned and the members of the population
ccc         selected for crossover.
          CALL REPLI(I,M)
          CALL REPLI(I,N)

Ccc  PROBABILITY OF CROSSOVER SET TO P_C = 0.9
Ccc  START CROSS OVER OPERATER
         LL = 1
 103      II = NINT(NV*LEN*URAND(ISEED)+0.5)
          DO 69 K = 1,II
```

```
            KK = K
            TGENO1(1,KK) = GENO1(N,KK)
            TGENO1(2,KK) = GENO1(M,KK)
 69       CONTINUE
          DO 68 K = II+1,NV*LEN
            KK = K
            TGENO1(1,KK) = GENO1(M,KK)
            TGENO1(2,KK) = GENO1(N,KK)
 68       CONTINUE

Ccc start MUTATION operator
          DO 35 L = 1,2
            DO 35 K = 1,NV*LEN
              IF (URAND(ISEED)*MUTATE.LE.1.0) THEN
                TGENO1(L,K)=ABS(TGENO1(L,K)-1.)
              ENDIF
 35       CONTINUE

Ccc  assign temporary binary string to array for GEN x+1
          DO 350 K = 1,NV*LEN
            GENO2(I,K) = TGENO1(1,K)
 350        GENO2(I+1,K) = TGENO1(2,K)

Ccc Convert new gene sequences to real values and determine corresponding
ccc objective values.
          DO 43 JJ = I,I+1
            CALL VALS(JJ)
            DO 411 K = 1,NV
 411          PHENO2(JJ,K) = POPY(K)
            CALL FX(JJ)
 43       CONTINUE
C
 97   CONTINUE

CCCC
```

```
        DO 530 K = 1,NV
 530      OUT(J,K+NFNCT) = PHENO1(J,K)
 510  CONTINUE
      CALL OUTFILE(IPROG)

      RETURN
      END
************************************************************************
************************************************************************
      REAL FUNCTION PHCOUNTER(N,MAX)
************************************************************************
************************************************************************
        INCLUDE 'PARAMSGA'
        INCLUDE 'PARAMSOBJ'
        INCLUDE 'COMMONSOBJ'
        INCLUDE 'COMMONSGA'

Ccc determines how crowded GenX member N would be if place into the
ccc partially filled next generation of size MAX.
      SCALE = 0.01
      PHCOUNTER = 0.
      DO 700 I=1,MAX-1,1
        D = 0.0
        DO 701 J = 1,NV
          DIFF = (PHENO2(I,J)-PHENO1(N,J))/DIFFER(J)
          D = D + DIFF*DIFF
 701    CONTINUE
        D = SQRT(D)
        IF (D.LE.SCALE) THEN
          PHCOUNTER = PHCOUNTER +1.0-D/SCALE
        ENDIF
 700  CONTINUE
      RETURN

      END
************************************************************************
************************************************************************
       SUBROUTINE VALS(JJ)
************************************************************************
************************************************************************
        INCLUDE 'PARAMSGA'
        INCLUDE 'PARAMSOBJ'
        INCLUDE 'COMMONSOBJ'
        INCLUDE 'COMMONSGA'

Ccc Determines the real value of binary encoded string for member JJ
      ICOUNT=1
      DO 43 K = 1,NV*LEN,LEN
       POPY(ICOUNT)= DECIMAL(ICOUNT,JJ)*RMAXMIN(1,ICOUNT)
     &                                  +RMAXMIN(2,ICOUNT)
       ICOUNT=ICOUNT+1
 43   CONTINUE
       RETURN
      END
************************************************************************
************************************************************************
      REAL FUNCTION DECIMAL(K,J)
************************************************************************
************************************************************************
        INCLUDE 'PARAMSGA'
        INCLUDE 'PARAMSOBJ'
        INCLUDE 'COMMONSOBJ'
        INCLUDE 'COMMONSGA'
Ccc  Determines the decimal equivalent of a binary encoded string.

      IALLEL = (K-1)*LEN
      DECIMAL = 0.0
      DO 74 I = LEN, 1, -1
      IF (INT(GENO2(J,IALLEL+I)).EQ.1) THEN
       DECIMAL=MAXLEN(LEN-I+1)+DECIMAL
      ENDIF
 74   CONTINUE
      RETURN
      END
************************************************************************
************************************************************************
      SUBROUTINE OUTFILE(IPROG)
************************************************************************
************************************************************************
        INCLUDE 'PARAMSGA'
        INCLUDE 'PARAMSOBJ'
```

```
        INCLUDE 'COMMONSOBJ'
        INCLUDE 'COMMONSGA'

CCCC  finds difference between the calculated and experimental values
        DO 77 N = 1,IPOPU
          DO 77 M = 1,NFNCT
            OUT(N,M) = (OUT(N,M)-FNCT(M))
 77     CONTINUE

ccc writes objective data and variable data to Tape7 and Tape8 respectively.
       DO 56 N = 1,IPOPU
          OUT(N,NV+NFNCT+1) = IPROG
 56     CONTINUE
        DO 65 N = 1,IPOPU
       WRITE(30,654)(OUT(N,I),I=1      ,NFNCT     ),OUT(N,1+NFNCT+NV)
       WRITE(40,654)(OUT(N,I),I=1+NFNCT,1+NFNCT+NV)
 65     CONTINUE
 654    FORMAT(1X,80(E11.4,' '))
C
        DO 67 N = 1,IPOPU
          DO 67 M = 1,1+NV+NFNCT
            OUT(N,M) = 0.
 67     CONTINUE
C
      RETURN
      END
*********************************************************************
*********************************************************************
        REAL FUNCTION URAND(IY)
*********************************************************************
*********************************************************************
        INTEGER IY
C   URAND IS A UNIFORM RANDOM NUMBER GENERATOR BASED ON THE THEORY AND
C   SUGGESTIONS GIVEN BY D.E. KNUTH ( REF.: 'SEMINUMERICAL ALGORITHMS' VOL. 2
C   MASS. READING ADDISON WESLEY 1969 ). THE INTEGER IY SHOULD BE INITIALIZED
C   PRIOR TO THE FIRST CALL TO AN ARBITRARY INTEGER. THE CALLING PROGRAM
C   SHOULD NOT ALTER THE VALUE OF IY BETWEEN SUBSEQUENT CALLS OF URAND.
C   VALUES OF URAND WILL BE RETURNED IN THE INTERVAL [0,1)
C
        INTEGER IA,IC,ITWO,M2,M,MIC
        DOUBLE PRECISION HALFM
        REAL S
C       DOUBLE PRECISON DATAN,DSQRT

        DATA M2/0/,ITWO/2/
        DATA IA,IC,MIC/0,0,0/
        DATA HALFM/0.0/

        IF(M2.NE.0) GOTO 20

C
C   IF FIRST ENTRY COMPUTE MACHINE INTEGER WORD LENGTH
C

        M=1
10      M2 = M
        M = ITWO*M2
        IF(M.GT.M2) GOTO 10
        HALFM = M2

C
C   COMPUTE MULTIPLIER AND INCREMENT FOR LINEAR CONGRUENTIAL METHOD
C

        IA = 8*IDINT(HALFM*DATAN(1.D0)/8.D0) + 5
        IC = 2*IDINT(HALFM*(0.5D0-DSQRT(3.D0)/6.D0)) + 1
        MIC = (M2 - IC) + M2

C
C   S IS THE SCALE FACTOR FOR CONVERTING TO A FLOATING POINT
C

        S = 0.5/HALFM

C
C   COMPUTE NEXT RANDOM NUMBER
C

20      IY = IY*IA

C
```

```
C   THE FOLLOWING STATEMENT IS FOR COMPUTERS WHICH DO NOT ALLOW
C   INTEGER OVERFLOW ON ADDITION
C

        IF (IY.GT.MIC) IY = (IY - M2) -M2
C
        IY = IY + IC
C
C   THE FOLLOWING STATEMENT IS FOR COMPUTERS WHERE THE
C   WORD LENGTH FOR ADDITION IS GREATER THAN FOR MULTIPLICATION
C

        IF(IY/2 .GT. M2 ) IY = (IY - M2) - M2

C
C   THE FOLLOWING STATEMENT IS FOR COMPUTERS WHERE THE INTEGER
C   OVERFLOW AFFECTS THE SIGN BIT
C

        IF (IY.LT.0) IY = (IY + M2) + M2
        URAND = FLOAT(IY)*S
        RETURN
        END
**********************************************************************
**********************************************************************
        SUBROUTINE GAINPUT_ROUTINE
**********************************************************************
**********************************************************************
        INCLUDE 'PARAMSGA'
        INCLUDE 'PARAMSOBJ'
        INCLUDE 'COMMONSOBJ'
        INCLUDE 'COMMONSGA'

Ccc READING THE HEADVAR FILE
        OPEN (UNIT=10,FILE='HEADVAR',STATUS='OLD')
        READ(10,*) NV
        DO 249 K = 1,NV
                READ(10,*) RMAXMIN(1,K), RMAXMIN(2,K)
 249    CONTINUE
       CLOSE(10)

Ccc READING THE HEADGA FILE
        OPEN (UNIT=10,FILE='HEADGA',STATUS='OLD')
        READ(10,*) LEN
        READ(10,*) IGEN
        READ(10,*) IPOPU
        READ(10,*) MUTATE
        READ(10,*) ICOMP
        CLOSE(10)
        RETURN
        END
**********************************************************************
**********************************************************************
      SUBROUTINE REPLI(MAX,MEM)
**********************************************************************
**********************************************************************
        INCLUDE 'PARAMSGA'
        INCLUDE 'PARAMSOBJ'
        INCLUDE 'COMMONSOBJ'
        INCLUDE 'COMMONSGA'
C
       LOGICAL D2,D1
       DIMENSION RMAX(MXNFNCT),RMIN(MXNFNCT)
Ccccccccccccccccccccccccccccccccccccccccccccccccc
ccc Randomly select two members of the population
Ccccccccccccccccccccccccccccccccccccccccccccccccc
       SCALER = 1.
 601   M = NINT(URAND(ISEED)*IPOPU+0.5)
 602   N = NINT(URAND(ISEED)*IPOPU+0.5)
Ccccccccccccccccccccccccccccccccccccccccccccccccc
ccc find the most unfit members of the population for the fuzzy logic selectionmethod
Ccccccccccccccccccccccccccccccccccccccccccccccccc
        DO 40 J=1,NFNCT
          RMAX(J) = FNCT(J)+ERROR(J)
          RMIN(J) = FNCT(J)-ERROR(J)
 40     CONTINUE
        DO 41 J = 1,IPOPU
        DO 41 I = 1,NFNCT
          IF (ATT1(J,I).GT.RMAX(I)) RMAX(I) = ATT1(J,I)
          IF (ATT1(J,I).LT.RMIN(I)) RMIN(I) = ATT1(J,I)
 41     CONTINUE
```

```
Ccccccccccccccccccccccccccccccccccccccccccccccccccc
        D1 = .TRUE.
        D2 = .TRUE.
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
cc For all objectives or functions determine the fuzzy fitness of
cc the two randomly selected members M and N
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
         AVERAGEM = 0.
         AVERAGEN = 0.
         DO 35 J = 1,NFNCT
            X1 = RMIN(J)
            Y1 = SCALER
            X2 = FNCT(J)-ERROR(J)
            Y2 = 0.0
            X3 = FNCT(J)
            Y3 = 0.0
            X4 = FNCT(J)+ERROR(J)
            Y4 = 0.0
            X5 = RMAX(J)
            Y5 = SCALER

            X = ATT1(M,J)
            AVERAGEM = AVERAGEM+AVG(X1,Y1,X2,Y2,X3,Y3,X4,Y4,X5,Y5,X)
            X = ATT1(N,J)
            AVERAGEN = AVERAGEN+AVG(X1,Y1,X2,Y2,X3,Y3,X4,Y4,X5,Y5,X)

 35      CONTINUE
         AVERAGEM = AVERAGEM/REAL(NFNCT)
         AVERAGEN = AVERAGEN/REAL(NFNCT)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C COMPARISON SET
cc For all objectives or functions determine the fuzzy fitness of
cc the members of the comparison set.
cc If the average of any comparison set member is less than the average
cc of M then D1 is set FALSE.  Likewise, D2 is set to FALSE if
cc the fuzzy fitness of any comparison set member is less than that of N.
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        DO 603 I = 1,ICOMP

 604     II = NINT(URAND(ISEED)*IPOPU+0.5)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
         AVERAGEO= 0.
         DO 36 J = 1,NFNCT
            X1 = RMIN(J)
            Y1 = SCALER
            X2 = FNCT(J)-ERROR(J)
            Y2 = 0.0
            X3 = FNCT(J)
            Y3 = 0.0
            X4 = FNCT(J)+ERROR(J)
            Y4 = 0.0
            X5 = RMAX(J)
            Y5 = SCALER

            X = ATT1(II,J)
            AVERAGEO = AVERAGEO+AVG(X1,Y1,X2,Y2,X3,Y3,X4,Y4,X5,Y5,X)

 36      CONTINUE
         AVERAGEO = AVERAGEO/REAL(NFNCT)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
         IF (AVERAGEO.LT.AVERAGEM) D1=.FALSE.
         IF (AVERAGEO.LT.AVERAGEN) D2=.FALSE.
 603    CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCC here is a tricky part.
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
ccc IF D1 is equal to D2 (they are both true or both false)
ccc then niching is needed.
cccccccccccccccccccccccccccccccccccccccccccccccccccc
       IF (D1.EQ.D2) THEN
ccc   count up the crowding factor for M and N
              RN1 =  PHCOUNTER(M,MAX)
              RN2 =  PHCOUNTER(N,MAX)
              IF (RN1.LT.RN2) THEN
                     MEM=M
              ELSEIF (RN2.LT.RN1) THEN
                     MEM=N
              ELSEIF (RN1.EQ.RN2) THEN
                     IF (URAND(ISEED).LE.0.5) THEN
```

```
                                MEM=M
                        ELSE
                                MEM=N
                        ENDIF
                ENDIF
        ELSEIF (D2) THEN
                MEM=N
        ELSEIF (D1) THEN
                MEM=M
        ENDIF
        RETURN
        END
***********************************************************************
***********************************************************************
      REAL FUNCTION AVG(X1,Y1,X2,Y2,X3,Y3,X4,Y4,X5,Y5,X)
***********************************************************************
***********************************************************************
ccc  This function is the heart of the fuzzy logic selection method.
ccc  The points listed above define the five major points in a
ccc  fuzzy logic rule set:
ccc  1\           /5
ccc    \         /
ccc     \       /
ccc      2__3__4
ccc AVG determines between which two points the value of X lies
ccc and then calls POINT to determine the cooresponding value of Y
cccccccccccccccccccccccccccccccccccccccccccccccccc
           IF (X1.GT.X2) THEN
            PRINT*,'ERROR,X1,X2'
            STOP
           ENDIF
          IF (X2.GT.X3) THEN
            PRINT*,'ERROR,X2,X3'
            STOP
           ENDIF
          IF (X3.GT.X4) THEN
            PRINT*,'ERROR,X3,X4'
            STOP
           ENDIF
          IF (X4.GT.X5) THEN
            PRINT*,'ERROR,X4,X5'
            STOP
           ENDIF

           IF (X.LE.X2) THEN
             AVG = POINT(X1,Y1,X2,Y2,X)
            ELSEIF (X.LE.X3) THEN
             AVG = POINT(X2,Y2,X3,Y3,X)
            ELSEIF (X.GE.X4) THEN
             AVG = POINT(X4,Y4,X5,Y5,X)
            ELSEIF (X.GE.X3) THEN
             AVG = POINT(X3,Y3,X4,Y4,X)
            ELSE
             PRINT*,'ERROR,OUT OF RANGE'
             STOP
            ENDIF
      RETURN
      END
***********************************************************************
***********************************************************************
      REAL FUNCTION POINT(X1,Y1,X2,Y2,X)
***********************************************************************
***********************************************************************
ccc The function point sends back a value of Y given two points on a line and a value of X.
ccc
      IF (X1.EQ.X2) THEN
       POINT = Y1
      ELSE
        SLOPE = (Y1-Y2)/(X1-X2)
        B  = Y1-SLOPE*X1
        POINT = SLOPE*X+B
      ENDIF
      RETURN
      END
```