1 SUMMARY

This package uses the genetic algorithm to search for a small value of an objective function of n binary (zero-one) variables. Each *string* of binary variables is stored in a logical array. A *population* of p such strings is maintained along with their associated objective function values. The population evolves in a sequence of iterations. The best features of the population at iteration k are passed to to the population at iteration k+1 by means of *mutation* and *crossover*.

The package obtains function values by reverse communication. The user is periodically required to check for termination.

ATTRIBUTES — Version: 1.0.0. **Types:** Real (single, double). **Calls:** HSL_FA14, HSL_ZA03. **Original date:** September 1995. **Origin:** N. I. M. Gould, Rutherford Appleton Laboratory. **Language:** Fortran 90.

2 HOW TO USE THE PACKAGE

Access to the package requires a USE statement such as

Single precision version

```
USE HSL_VH01_SINGLE
```

Double precision version

```
USE HSL_VH01_DOUBLE
```

If it is required to use both modules at the same time, the derived types VH01_DIMENSION, VH01_PARAMETERS, VH01_STRING VH01_EVALUATE, and VH01_INFORMATION (Section 2.1), and the subroutines VH01_INITIALIZE, VH01_ITERATION and VH01_WIND_UP (Section 2.2) must be renamed on one of the USE statements.

2.1 The derived data types

Five publically accessible derived data types are used by the package.

2.1.1 The derived data type for holding problem dimensions

The derived data type VH01_DIMENSION is used to hold problem dimensions. The components of VH01_DIMENSION are:

n is a scalar variable of type default INTEGER which holds the number of binary variables, n.

p is a scalar variable of type default INTEGER which holds the size of the population, p.

m is a scalar variable of type default INTEGER which holds the number, m, of pairs in the population that will be subjected to crossover (see Section 4).

2.1.2 The derived data type for holding algorithmic parameters

The derived data type VH01_PARAMETERS is used to hold parameters which control the minimization. The components of VH01_PARAMETERS are:

xi is a scalar variable of type default REAL (double precision REAL in HSL_VH01_DOUBLE) which holds the crossover probability, ξ (see section 4).

mu is a scalar variable of type default REAL (double precision REAL in HSL_VH01_DOUBLE) which holds the mutation probability, μ (see section 4).

twopt is a scalar variable of type LOGICAL(KIND=ZA03_1BYTE) (see the package HSL_ZA03 for details) which specifies whether two-point crossover is to be used (see section 4).

wrap is a scalar variable of type LOGICAL(KIND=ZA03_1BYTE) (see the package HSL_ZA03 for details) which specifies whether the crossover is to be allowed to wrap around (see section 4).

2.1.3 The derived data type for holding population information

The derived data type VH01_STRING is used to hold an estimate of the minimizer together with its associated objective function value. The components of VH01_STRING are:

x is a rank-one pointer array of type LOGICAL(KIND=ZA03_1BYTE) (see the package HSL_ZA03 for details) which represents a string of binary variables. The *i*–th component of x has the value .TRUE. if the *i*–th component of the string is 1, and is .FALSE. otherwise.

f is a scalar variable of type default REAL (double precision REAL in HSL_VH01_DOUBLE) which contains the value of the objective function corresponding to the string represented in x.

2.1.4 The derived data type for recording population changes

The derived data type VH01_EVALUATE is used to define which strings have changed, and thus which objective function values need to be recomputed. The components of VH01_EVALUATE are:

changes is a scalar variable of type default INTEGER which holds the number of members of the population that have changed.

list is a pointer array of type default INTEGER which holds a list of the members of the population that have changed.

seed is a scalar variable of type FA14_SEED which holds the generator word used in calls by HSL_VH01 to members of the random number package HSL_FA14. It is set to the same initial value (to generate the same random sequence) each time VH01_INITIALIZE is called. This behaviour may be controlled by appropriate use of the FA14_GET_SEED and FA14_SET_SEED entries before and after the initialization calls.

2.1.5 The derived data type for holding algorithmic information

The derived data type VH01_INFORMATION is used to hold information concerning the progress of the minimization. The components of VH01_INFORMATION are:

inform is a scalar variable of type default INTEGER which controls the action of the package.

iter is a scalar variable of type default INTEGER which holds the number of iterations performed.

best is a scalar variable of type default INTEGER which holds the number of the individual in the current population that yields the smallest objective value.

warnings is a scalar variable of type default INTEGER which holds the unit number for any warning messages issued. A non-positive value of warnings stops warning messages.

errors is a scalar variable of type default INTEGER which holds the unit number for any error messages issued. A non-positive value of errors stops error messages.

progress is a scalar variable of type default INTEGER which holds the unit number for any messages about the progress of the minimization. A non-positive value of progress stops these messages.

2.2 Argument lists and calling sequences

There are three procedures for user calls:

1. The subroutine VH01_INITIALIZE is used to set default values and initialize allocatable array dimensions. This procedure must be called prior to the minimization.
2. The subroutine VH01_ITERATION is called repeatedly to perform the minimization. On each exit, the user is expected to provide additional information and, if necessary, re-enter the subroutine.

3. The subroutine VH01_WIND_UP is provided to allow the user to automatically deallocate the arrays allocated in VH01_INITIALIZE at the end of the minimization. This subroutine should be used before another minimization is attempted.

2.2.1 The initialization subroutine

Default values are provided and arrays allocated as follows:

```
CALL VH01_INITIALIZE( DIMEN, POP, PARAM, EVAL, INFO )
```

DIMEN is a scalar INTENT(INOUT) argument of type VH01_DIMENSIONS. The components n, p and m must be set to the number of binary variables *n*, the size of the population *p*, and the number of pairs of crossovers *m*, respectively. If m is less than 1, it will be reset to 1, and if it is greater than $(p-1)/2$, it will be reset to $(p-1)/2$. Otherwise, DIMEN will not be altered on output. **Restriction:** DIMEN%n ≥ 1, DIMEN%p ≥ 1.

POP is an array INTENT(OUT) argument of type VH01_STRING and dimension PARAM%p which need not be set on input. On exit, the component x for each member of the array will have been allocated with length DIMEN

PARAM is a scalar INTENT(OUT) argument of type VH01_PARAMETERS which need not be set on input. On output, PARAM contains default values for the components xi $=0.9$, mu $=0.5$, twopt = .TRUE., and wrap = .TRUE.. These values should only be changed after calling VH01_INITIALIZE.

EVAL is a scalar INTENT(OUT) argument of type VH01_EVALUATE which need not be set on input. On exit, the component list will have been allocated with length DIMEN

INFO is a scalar INTENT(INOUT) argument of type VH01_INFORMATION. The components warnings, errors, and progress should be set to the unit numbers for warning, error and information messages. A non-positive value suppresses the appropriate class of message. The remaining components need not be set. A successful call to VH01_INITIALIZE is indicated when the component inform has the value 0. For other return values of inform, see Section 2.4

2.2.2 The minimization subroutine

The genetic algorithm is called as follows:

```
CALL VH01_ITERATION( DIMEN, POP, PARAM, EVAL, INFO )
```

DIMEN is a scalar INTENT(IN) argument of type VH01_DIMENSIONS. DIMEN should be unaltered since the last call to VH01_INITIALIZE.

POP is an array INTENT(INOUT) argument of type VH01_STRING and dimension PARAM%p. On initial (INFO $1 \leq$ i $\leq$ PARAM%p, should be set to an estimate of the minimizing binary string. The *j*–th component of POP(i) value 1, and .FALSE. if the variable has the value 0. It is preferable if the selected initial strings are different, but this is not crucial. The component POP(i) corresponding to the string in POP(i) On each subsequent exit, POP(i) the minimizing binary string. The user will be directed to re-evaluate POP(i) INFO

PARAM is a scalar INTENT(IN) argument of type VH01_PARAMETERS which must contain values for the components xi, mu, twopt and wrap. These values will have been assigned defaults by VH01_INITIALIZE. **Restriction:** $0 \leq$ PARAM%xi ≤ 1, $0 \leq$ PARAM%mu ≤ 1.

EVAL is a scalar INTENT(INOUT) argument of type VH01_EVALUATE which need not be set on initial (INFO the first EVAL%changes components of EVAL%list contain the indices of members of the population that have changed during the current iteration.

INFO is a scalar INTENT(INOUT) argument of type VH01_INFORMATION. The component inform must be unaltered since the last successful call to VH01_INITIALIZE, and will subsequently be reset on exit from VH01_ITERATION to indicate what action needs be taken by the user before re-entering the subroutine (see Section 2.3). The components warnings, errors, and progress should be unaltered since the last successful call to VH01_INITIALIZE and are unaltered by VH01_ITERATION. The components iter and best need not

be set on initial (INFO and on exit contain the current iteration number *k* and the index of the member of the population which currently provides the smallest objective function value respectively.

2.2.3 The termination subroutine

All previously allocated arrays are deallocated as follows:

```
CALL VH01_WIND_UP( DIMEN, POP, EVAL, INFO )
```

DIMEN is a scalar INTENT(IN) argument of type VH01_DIMENSIONS which must be passed unaltered from VH01_ITERATION.

POP is an array INTENT(INOUT) argument of type VH01_STRING and dimension PARAM%p which must be passed unaltered from VH01_ITERATION. The component x of each member of the array will have been deallocated on exit.

EVAL is a scalar INTENT(INOUT) argument of type VH01_EVALUATE which must be passed unaltered from VH01_ITERATION. The component list will have been deallocated on exit.

INFO is a scalar INTENT(INOUT) argument of type VH01_INFORMATION which must be passed unaltered from VH01_ITERATION. A successful call to VH01_WIND_UP is indicated when the component inform has the value 0. For other return values of inform, see Section 2.4

2.3 Reverse communication

A negative value of INFO VH01_ITERATION indicates that the user needs to take appropriate action before re-entering the subroutine. Possible values are:

–1. The objective values of the strings indexed by EVAL%list(i), i=1,...,EVAL%changes, must be recomputed. The value for string POP(j)_x must be formed in POP(j)_f. All other arguments must be left unaltered and VH01_ITERATION re-entered.

–2. The current smallest objective value occurs for string INFO_best at the end of iteration INFO_iter. The user may check for termination at this stage. If convergence has not occurred, all arguments must be left unaltered and VH01_ITERATION re-entered prior to another iteration. The user may wish to consider whether excessive iterations have occurred, whether the objective function value has reached a known limit, or if some other measure of convergence has been achieved.

2.4 Warning and error messages

A positive value of INFO VH01_INITIALIZE or VH01_WIND_UP indicates that an error has occurred. No further calls should be made until the problem has been resolved. Possible values are:

1. (VH01_INITIALIZE only) An array allocation has failed. A message indicating the offending array is written on unit INFO%errors.
2. (VH01_WIND_UP only) An array deallocation has failed. A message indicating the offending array is written on unit INFO%warnings.

2.6 Information printed

If INFO%progress is positive, information about the progress of the minimization will be printed on unit INFO%progress. Each time an improvement in objective function is made, a line indicating the iteration on which the event occurs, the value of the function and the corresponding binary string is printed.

3 GENERAL INFORMATION

Use of common: None.

Other modules used directly: HSL_FA14, HSL_ZA03.

Input/output: Output is under control of the arguments `INFO%warnings`, `INFO%errors`, and `INFO%progress`.

Restrictions: `DIMEN%n` ≥ 1, `DIMEN%p` ≥ 1, $0 \leq$ `PARAM%xi` ≤ 1, $0 \leq$ `PARAM%mu` ≤ 1.

4 METHOD

There are many variants of the genetic algorithm. The version implemented in `HSL_VH01` is as follows.

An initial population of (preferably different) binary strings $\{x_1^{(0)},...,x_p^{(0)}\}$ is specified, and the value of the objective function $F_1^{(0)},...,F_p^{(0)}$ for each member of the population is calculated. An iteration counter k is initialized as 1. We describe the k–th iteration.

A *probability* function $p_i^{(k)}$ is associated with each member of the current population. In `HSL_VH01`, this probability is given as

$$p_i^{(k)} = \frac{F_{\max}^{(k)} - F_i^{(k)}}{\sum_{i=1}^{p} (F_{\max}^{(k)} - F_i^{(k)})},$$

where $F_{\max}^{(k)} = \max_{1 \leq i \leq p} F_i^{(k)}$. The iteration comprises four essential steps, namely selection, crossover, mutation and substitution. Many sophisticated crossover and mutation strategies have been suggested, and we provide simple versions here.

In the selection step, an ordered set of $2m$ different individuals, $\{y_1^{(k)},...,y_p^{(k)}\}$, from the current population are selected. Each is selected as follows. A random integer i in $[1,p]$ is determined. If i has not already been selected, it is selected provided that $p_i^{(k)} > \phi$, where ϕ is a uniform random number between zero and one.

In the crossover step, each pair $\{y_{2j}^{(0)},...,y_{2j+1}^{(k)}\}$ is *crossed over* provided that $\phi > \xi$, where ξ is a real constant in [0,1], and once again ϕ is a uniform random number between zero and one. This yields m pairs of children $\{z_{2j}^{(0)},...,z_{2j+1}^{(k)}\}$ (which are identical to their parents with probability $1 - \xi$). We allow two crossover strategies. In *two-point* crossover, two positions, l_1 and l_2 are chosen at random; in *one-point*, a single random position l_1 is selected, and l_2 is set to p. The pair of strings x and y are now crossed over, that is components between l_1 and l_2 of x and y are interchanged. If *wrap around* is allowed, and $l_1 > l_2$, components l_1 to p and 1 to l_2 of x and y are interchanged instead of l_2 to l_1.

In the mutation step, each individual $z_i^{(k)}$ mutates provided that $\phi > \mu$, where μ is another real constant in [0,1], and once again ϕ is a uniform random number between zero and one. This provides $2m$ possibly mutated children $w_i^{(k)}$ (which are identical to their parents $z_i^{(k)}$ with probability $1 - \mu$).

Our mutation strategy is very simple. A position l is chosen at random. The component of the mutating string in position l switches parity, that is, if position l contains a zero, its value is changed to one, and vice versa.

Finally, in the substitution step, another ordered set of $2m$ different individuals from the current population are selected. Each is selected as follows. A random integer i in $[1,p]$ is determined. If i has not already been selected, it is selected provided that $p_i^{(k)} < \phi$, where once again ϕ is a uniform random number between zero and one. The individual which gives the lowest function value amongst the k–th population is also excluded. The $2m$ individuals in the selected set are then replaced in the population by the $2m$ crossed-over, mutated individuals $w_i^{(k)}$ to form the $k+1$–st population $\{x_1^{(k+1)},...,x_p^{(k+1)}\}$. The values of the objective function, $F_1^{(k+1)},...,F_p^{(k+1)}$, for the new population are calculated, and the iteration counter k incremented by one.

Reference

There are many references on the subject. A good, simple introduction is provided by

Pirlot, M. (1992). General local search heuristics in combinatorial optimization: a tutorial, Belgium Journal of Operations Research, Statistics and Computer Science, **32,** 7-68.

5 EXAMPLE

Suppose we wish minimize a function of 48 binary variables, whose value is the sum of the these variables, and that we wish to continue so long as the function is positive. Then we might use the following code:

```
      PROGRAM HSL_VH01_SPEC
      USE HSL_FA14_DOUBLE, ONLY: FA14_RANDOM_INTEGER
      USE HSL_VH01_DOUBLE
      IMPLICIT NONE
      INTEGER, PARAMETER :: length_X = 48 ! length of string
      INTEGER, PARAMETER :: npop = 100 !  population size
      INTEGER, PARAMETER :: mcross = 10 ! max. number crossovers/generation
      INTEGER, PARAMETER :: itmax  = 1000000 !  max number of generations
      INTEGER :: i, j, random_integer
      TYPE ( VH01_DIMENSIONS ) :: DIMEN
      TYPE ( VH01_STRING ), DIMENSION( npop ) :: POP
      TYPE ( VH01_PARAMETERS ) :: PARAM
      TYPE ( VH01_EVALUATE ) :: EVAL
      TYPE ( VH01_INFORMATION ) :: INFO
      DIMEN%n = length_X ; DIMEN%p = npop ; DIMEN%m = mcross ! set dimensions
      INFO%warnings = 6 ; INFO%errors = 6 ; INFO%progress = 6 ! set i/o units
      CALL VH01_INITIALIZE( DIMEN, POP, PARAM, EVAL, INFO )
      IF ( INFO%inform > 0 ) GO TO 100
      DO j = 1, DIMEN%p    !  Select an initial population x1, ... , xN
         DO i = 1, DIMEN%n
            CALL FA14_RANDOM_INTEGER( EVAL%seed, 2, random_integer )
            POP( j )%x( i ) = random_integer == 2
         END DO
         POP( j )%f = COUNT( POP( j )%x )   ! Evaluate the objective function
      END DO
      DO   !  start of main iteration
         CALL VH01_ITERATION( DIMEN, POP, PARAM, EVAL, INFO )
         IF ( INFO%inform == - 1 ) THEN   !  Evaluate the objective function
            DO i = 1, EVAL%changes
               j = EVAL%list( i )
               POP( j )%f = COUNT( POP( j )%x )
            END DO
            CYCLE
         ELSE IF ( INFO%inform == - 2 ) THEN   !  Check for termination
            IF ( INFO%iter <= itmax .AND.                                  &
                 POP( INFO%best )%f > 0.0_working ) CYCLE
            IF ( INFO%iter > itmax ) THEN ; INFO%inform = 3
            ELSE ; INFO%inform = 0
            END IF
         ENDIF
         EXIT
      END DO    !  end of main iteration
  100 CONTINUE
      IF ( INFO%inform == 0 ) THEN
         WRITE( 6, "( /, ' best objective ', ES12.4, /, ' string      ',    &
                   &( 80I1 ) ) " ) POP( INFO%best )%f,                     &
                                   MERGE( 1, 0, POP( INFO%best )%x( : ) )
      ELSE
         WRITE( 6, "(' Error exit: inform = ', I6 )" ) INFO%inform
      END IF
      CALL VH01_WIND_UP( DIMEN, POP, EVAL, INFO )
      END PROGRAM HSL_VH01_SPEC
```

This produces the following output:

```
 It     1 obj  1.80E+01 str 00010101010100100011110011101010000010110000000
 It     4 obj  1.40E+01 str 01101010010000010001000100100010000100000001011
 It    14 obj  1.30E+01 str 01101010010000010001000000100010000100000001101
 It    17 obj  1.20E+01 str 01101010010000010001000000100010000100000001100
 It    20 obj  9.00E+00 str 00000010010000010001000000100010000100000001100
 It    28 obj  8.00E+00 str 00000010010000000001000000100010000100000001100
 It    45 obj  7.00E+00 str 00000010010000010001000000000010000100000001000
 It    58 obj  6.00E+00 str 00000010010000010000000000000010000100000001000
 It    74 obj  5.00E+00 str 00000010010000010000000000000010000100000000000
 It    81 obj  4.00E+00 str 00000010010000010000000000000000000100000000000
 It    99 obj  3.00E+00 str 00000010000000010000000000000010000000000000000
 It   107 obj  2.00E+00 str 00000010000000010000000000000000000000000000000
 It   126 obj  1.00E+00 str 00000000000000010000000000000000000000000000000
 It   193 obj  0.00E+00 str 00000000000000000000000000000000000000000000000

  best objective   0.0000E+00
  string      00000000000000000000000000000000000000000000000
```