NAG Library Routine Document

H05ABF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

Given a set of m features and a scoring mechanism for any subset of those features, H05ABF selects the best n subsets of size p using a direct communication branch and bound algorithm.

2 Specification

```
SUBROUTINE H05ABF (MINCR, M, IP, NBEST, LA, BSCORE, BZ, F, MINCNT,          &
                   GAMMA, ACC, IUSER, RUSER, IFAIL)
INTEGER            MINCR, M, IP, NBEST, LA, BZ(M-IP,NBEST), MINCNT,         &
                   IUSER(*), IFAIL
REAL (KIND=nag_wp) BSCORE(NBEST), GAMMA, ACC(2), RUSER(*)
EXTERNAL           F
```

3 Description

Given $\Omega = \{x_{\mathrm{i}} : i \in \mathbb{Z}, 1 \leq i \leq m\}$, a set of m unique features and a scoring mechanism $f(S)$ defined for all $S \subseteq \Omega$ then H05ABF is designed to find $S_{o1} \subseteq \Omega, |S_{o1}| = p$, an optimal subset of size p. Here $|S_{o1}|$ denotes the cardinality of S_{o1}, the number of elements in the set.

The definition of the optimal subset depends on the properties of the scoring mechanism, if

$$f(S_i) \leq f(S_j), \quad \text{for all } S_j \subseteq \Omega \text{ and } S_i \subseteq S_j \tag{1}$$

then the optimal subset is defined as one of the solutions to

$$\underset{S \subseteq \Omega}{\text{maximize}} f(S) \quad \text{subject to} \quad |S| = p$$

else if

$$f(S_i) \geq f(S_j), \quad \text{for all } S_j \subseteq \Omega \text{ and } S_i \subseteq S_j \tag{2}$$

then the optimal subset is defined as one of the solutions to

$$\underset{S \subseteq \Omega}{\text{minimize}} f(S) \quad \text{subject to} \quad |S| = p.$$

If neither of these properties hold then H05ABF cannot be used.

As well as returning the optimal subset, S_{o1}, H05ABF can return the best n solutions of size p. If S_{oi} denotes the ith best subset, for $i = 1, 2, \ldots, n-1$, then the $(i+1)$th best subset is defined as the solution to either

$$\underset{S \subseteq \Omega - \{S_{oj} : j \in \mathbb{Z}, 1 \leq j \leq i\}}{\text{maximize}} f(S) \quad \text{subject to} \quad |S| = p$$

or

$$\underset{S \subseteq \Omega - \{S_{oj} : j \in \mathbb{Z}, 1 \leq j \leq i\}}{\text{minimize}} f(S) \quad \text{subject to} \quad |S| = p$$

depending on the properties of f.

The solutions are found using a branch and bound method, where each node of the tree is a subset of Ω. Assuming that (1) holds then a particular node, defined by subset S_i, can be trimmed from the tree if $f(S_i) < \hat{f}(S_{on})$ where $\hat{f}(S_{on})$ is the nth highest score we have observed so far for a subset of size p, i. e., our current best guess of the score for the nth best subset. In addition, because of (1) we can also

drop all nodes defined by any subset S_j where $S_j \subseteq S_i$, thus avoiding the need to enumerate the whole tree. Similar short cuts can be taken if (2) holds. A full description of this branch and bound algorithm can be found in Ridout (1988).

Rather than calculate the score at a given node of the tree H05ABF utilizes the fast branch and bound algorithm of Somol *et al.* (2004), and attempts to estimate the score where possible. For each feature, x_i, two values are stored, a count c_i and $\hat{\mu}_i$, an estimate of the contribution of that feature. An initial value of zero is used for both c_i and $\hat{\mu}_i$. At any stage of the algorithm where both $f(S)$ and $f(S - \{x_i\})$ have been calculated (as opposed to estimated), the estimated contribution of the feature x_i is updated to

$$\frac{c_i\hat{\mu}_i + \left[f(S) - f\left(S - \left\{x_j\right\}\right)\right]}{c_i + 1}$$

and c_i is incremented by 1, therefore at each stage $\hat{\mu}_i$ is the mean contribution of x_i observed so far and c_i is the number of observations used to calculate that mean.

As long as $c_i \geq k$, for the user-supplied constant k, then rather than calculating $f(S - \{x_i\})$ this routine estimates it using $\hat{f}(S - \{x_i\}) = f(S) - \gamma\hat{\mu}_i$ or $\hat{f}(S) - \gamma\hat{\mu}_i$ if $f(S)$ has been estimated, where γ is a user-supplied scaling factor. An estimated score is never used to trim a node or returned as the optimal score.

Setting $k = 0$ in this routine will cause the algorithm to always calculate the scores, returning to the branch and bound algorithm of Ridout (1988). In most cases it is preferable to use the fast branch and bound algorithm, by setting $k > 0$, unless the score function is iterative in nature, i.e., $f(S)$ must have been calculated before $f(S - \{x_i\})$ can be calculated.

H05ABF is a direct communication version of H05AAF.

4 References

Narendra P M and Fukunaga K (1977) A branch and bound algorithm for feature subset selection *IEEE Transactions on Computers* **9** 917–922

Ridout M S (1988) Algorithm AS 233: An improved branch and bound algorithm for feature subset selection *Journal of the Royal Statistics Society, Series C (Applied Statistics) (Volume 37)* **1** 139–147

Somol P, Pudil P and Kittler J (2004) Fast branch and bound algorithms for optimal feature selection *IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume 26)* **7** 900–912

5 Arguments

1: MINCR – INTEGER *Input*

On entry: flag indicating whether the scoring function f is increasing or decreasing.

MINCR $= 1$
$f(S_i) \leq f\left(S_j\right)$, i.e., the subsets with the largest score will be selected.

MINCR $= 0$
$f(S_i) \geq f\left(S_j\right)$, i.e., the subsets with the smallest score will be selected.

For all $S_j \subseteq \Omega$ and $S_i \subseteq S_j$.

Constraint: MINCR $= 0$ or 1.

2: M – INTEGER *Input*

On entry: m, the number of features in the full feature set.

Constraint: M ≥ 2.

3: IP – INTEGER *Input*

On entry: p, the number of features in the subset of interest.

Constraint: $1 \leq \text{IP} \leq \text{M}$.

4: NBEST – INTEGER *Input*

On entry: n, the maximum number of best subsets required. The actual number of subsets returned is given by LA on final exit. If on final exit $\text{LA} \neq \text{NBEST}$ then $\text{IFAIL} = 42$ is returned.

Constraint: $\text{NBEST} \geq 1$.

5: LA – INTEGER *Output*

On exit: the number of best subsets returned.

6: BSCORE(NBEST) – REAL (KIND=nag_wp) array *Output*

On exit: holds the score for the LA best subsets returned in BZ.

7: BZ(M − IP, NBEST) – INTEGER array *Output*

On exit: the jth best subset is constructed by dropping the features specified in $\text{BZ}(i,j)$, for $i = 1,2,\ldots,\text{M} - \text{IP}$ and $j = 1,2,\ldots,\text{LA}$, from the set of all features, Ω. The score for the jth best subset is given in $\text{BSCORE}(j)$.

8: F – SUBROUTINE, supplied by the user. *External Procedure*

F must evaluate the scoring function f.

The specification of F is:

```
SUBROUTINE F (M, DROP, LZ, Z, LA, A, SCORE, IUSER, RUSER, INFO)
INTEGER            M, DROP, LZ, Z(LZ), LA, A(LA), IUSER(*), INFO
REAL (KIND=nag_wp) SCORE(max(LA,1)), RUSER(*)
```

1: M – INTEGER *Input*

On entry: $m = |\Omega|$, the number of features in the full feature set.

2: DROP – INTEGER *Input*

On entry: flag indicating whether the intermediate subsets should be constructed by dropping features from the full set ($\text{DROP} = 1$) or adding features to the empty set ($\text{DROP} = 0$). See SCORE for additional details.

3: LZ – INTEGER *Input*

On entry: the number of features stored in Z.

4: Z(LZ) – INTEGER array *Input*

On entry: $\text{Z}(i)$, for $i = 1,2,\ldots,\text{LZ}$, contains the list of features which, along with those specified in A, define the subsets whose score is required. See SCORE for additional details.

5: LA – INTEGER *Input*

On entry: if $\text{LA} > 0$, the number of subsets for which a score must be returned.

If $\text{LA} = 0$, the score for a single subset should be returned. See SCORE for additional details.

6: A(LA) – INTEGER array *Input*

On entry: A(j), for $j = 1, 2, \ldots, \text{LA}$, contains the list of features which, along with those specified in Z, define the subsets whose score is required. See SCORE for additional details.

7: SCORE(max(LA, 1)) – REAL (KIND=nag_wp) array *Output*

On exit: the value $f\left(S_j\right)$, for $j = 1, 2, \ldots, \text{LA}$, the score associated with the jth subset. S_j is constructed as follows:

DROP $= 1$
: S_j is constructed by *dropping* the features specified in the first LZ elements of Z and the single feature given in A(j) from the full set of features, Ω. The subset will therefore contain $\text{M} - \text{LZ} - 1$ features.

DROP $= 0$
: S_j is constructed by *adding* the features specified in the first LZ elements of Z and the single feature specified in A(j) to the empty set, $\emptyset$. The subset will therefore contain $\text{LZ} + 1$ features.

In both cases the individual features are referenced by the integers 1 to M with 1 indicating the first feature, 2 the second, etc., for some arbitrary ordering of the features, chosen by you prior to calling H05ABF. For example, 1 might refer to the first variable in a particular set of data, 2 the second, etc..

If LA $= 0$, the score for a single subset should be returned. This subset is constructed by adding or removing only those features specified in the first LZ elements of Z. If LZ $= 0$, this subset will either be Ω or $\emptyset$.

8: IUSER($*$) – INTEGER array *User Workspace*

9: RUSER($*$) – REAL (KIND=nag_wp) array *User Workspace*

F is called with the arguments IUSER and RUSER as supplied to H05ABF. You should use the arrays IUSER and RUSER to supply information to F.

10: INFO – INTEGER *Input/Output*

On entry: INFO $= 0$.

On exit: set INFO to a nonzero value if you wish H05ABF to terminate with IFAIL $= 82$.

F must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which H05ABF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

9: MINCNT – INTEGER *Input*

On entry: k, the minimum number of times the effect of each feature, x_i, must have been observed before $f\left(S - \{x_i\}\right)$ is estimated from $f\left(S\right)$ as opposed to being calculated directly.

If $k = 0$ then $f\left(S - \{x_i\}\right)$ is never estimated. If MINCNT < 0 then k is set to 1.

10: GAMMA – REAL (KIND=nag_wp) *Input*

On entry: γ, the scaling factor used when estimating scores. If GAMMA < 0 then $\gamma = 1$ is used.

11: ACC(2) – REAL (KIND=nag_wp) array *Input*

On entry: a measure of the accuracy of the scoring function, f.

Letting $a_i = \epsilon_1 |f\left(S_i\right)| + \epsilon_2$, then when confirming whether the scoring function is strictly increasing or decreasing (as described in MINCR), or when assessing whether a node defined by

subset S_i can be trimmed, then any values in the range $f(S_i) \pm a_i$ are treated as being numerically equivalent.

If $0 \leq \mathrm{ACC}(1) \leq 1$ then $\epsilon_1 = \mathrm{ACC}(1)$, otherwise $\epsilon_1 = 0$.

If $\mathrm{ACC}(2) \geq 0$ then $\epsilon_2 = \mathrm{ACC}(2)$, otherwise $\epsilon_2 = 0$.

In most situations setting both ϵ_1 and ϵ_2 to zero should be sufficient. Using a nonzero value, when one is not required, can significantly increase the number of subsets that need to be evaluated.

12: IUSER(∗) – INTEGER array *User Workspace*

13: RUSER(∗) – REAL (KIND=nag_wp) array *User Workspace*

IUSER and RUSER are not used by H05ABF, but are passed directly to F and should be used to pass information to this routine.

14: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value −1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this argument, the recommended value is 0. **When the value −1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL = 0 unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL = 11

On entry, MINCR = ⟨*value*⟩.
Constraint: MINCR = 0 or 1.

IFAIL = 21

On entry, M = ⟨*value*⟩.
Constraint: M ≥ 2.

IFAIL = 31

On entry, IP = ⟨*value*⟩ and M = ⟨*value*⟩.
Constraint: 1 ≤ IP ≤ M.

IFAIL = 41

On entry, NBEST = ⟨*value*⟩.
Constraint: NBEST ≥ 1.

IFAIL = 42

On entry, NBEST = ⟨*value*⟩.
But only ⟨*value*⟩ best subsets could be calculated.

IFAIL $= 81$

On exit from F, SCORE($\langle value \rangle$) $= \langle value \rangle$, which is inconsistent with the score for the parent node. Score for the parent node is $\langle value \rangle$.

IFAIL $= 82$

A nonzero value for INFO has been returned: INFO $= \langle value \rangle$.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

The subsets returned by H05ABF are guaranteed to be optimal up to the accuracy of the calculated scores.

8 Parallelism and Performance

H05ABF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The maximum number of unique subsets of size p from a set of m features is $N = \frac{m!}{(m-p)!p!}$. The efficiency of the branch and bound algorithm implemented in H05ABF comes from evaluating subsets at internal nodes of the tree, that is subsets with more than p features, and where possible trimming branches of the tree based on the scores at these internal nodes as described in Narendra and Fukunaga (1977). Because of this it is possible, in some circumstances, for more than N subsets to be evaluated. This will tend to happen when most of the features have a similar effect on the subset score.

If multiple optimal subsets exist with the same score, and NBEST is too small to return them all, then the choice of which of these optimal subsets is returned is arbitrary.

10 Example

This example finds the three linear regression models, with five variables, that have the smallest residual sums of squares when fitted to a supplied dataset. The data used in this example was simulated.

10.1 Program Text

```
!   H05ABF Example Program Text

!   Mark 26 Release. NAG Copyright 2016.
    Module h05abfe_mod

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                           :: f, prepare_user_arrays
!     .. Parameters ..
      Integer, Parameter, Public       :: nin = 5, nout = 6
    Contains
      Subroutine f(m,drop,lz,z,la,a,score,iuser,ruser,info)
!       Score calculating function required by H05ABF

!       M,DROP,LZ,Z,LA,A,SCORE IUSER and RUSER are all as described in the
!       documentation of H05ABF.

!       This particular example finds the set, of a given size, of explanatory
!       variables that best fit a response variable when a linear regression
!       model is used. Therefore the feature set is the set of all the
!       explanatory variables and the best set of features is defined as set
!       of explanatory variables that gives the smallest residual sums of
!       squares.
!       See the documentation for G02DAF for details on linear regression
!       models.

!       .. Use Statements ..
        Use nag_library, Only: g02daf
!       .. Scalar Arguments ..
        Integer, Intent (In)           :: drop, la, lz, m
        Integer, Intent (Inout)        :: info
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout) :: ruser(*)
        Real (Kind=nag_wp), Intent (Out) :: score(max(la,1))
        Integer, Intent (In)           :: a(la), z(lz)
        Integer, Intent (Inout)        :: iuser(*)
!       .. Local Scalars ..
        Real (Kind=nag_wp)             :: rss, tol
        Integer                        :: ex, ey, i, idf, ifail, inv_drop, ip, &
                                          irank, ldq, ldx, n, sx, sy
        Logical                        :: svd
        Character (1)                  :: mean, weight
!       .. Local Arrays ..
        Real (Kind=nag_wp), Allocatable :: b(:), cov(:), h(:), p(:), q(:,:),   &
                                          res(:), se(:), wk(:), wt(:)
        Integer, Allocatable           :: isx(:)
!       .. Intrinsic Procedures ..
        Intrinsic                      :: abs, count, max
!       .. Executable Statements ..
        Continue
        info = 0

        n = iuser(1)
        ldq = n
        ldx = n

!       No intercept term and no weights
        mean = 'Z'
        weight = 'U'

!       Allocate various arrays required by G02DAF
        Allocate (b(m),cov((m*m+m)/2),h(n),p(m*(m+2)),q(ldq,m+1),res(n),se(m), &
          wk(m*m+5*(m-1)),wt(1),isx(m))

!       Set up the initial feature set. If DROP = 0, this is the Null set
```

```
!       (i.e. no features). If DROP = 1 then this is the full set (i.e. all
!       features)
        isx(1:m) = drop

!       Add (if DROP = 0) or remove (if DROP = 1) the all the features
!       specified in Z
        inv_drop = abs(drop-1)
        Do i = 1, lz
          isx(z(i)) = inv_drop
        End Do

!       Get the start and end of X and Y in RUSER
        sx = iuser(4)
        ex = sx + m*n - 1
        sy = iuser(3)
        ey = sy + n - 1

!       Extract some parameters from RUSER
        tol = ruser(iuser(2))

        Do i = 1, max(la,1)
          If (la>0) Then
            If (i>1) Then
!             Reset the feature altered at the last iteration
              isx(a(i-1)) = drop
            End If

!           Add or drop the I'th feature in A
            isx(a(i)) = inv_drop
          End If

          ip = count(isx(1:m)==1)

!         Fit the regression model
          ifail = 0
          Call g02daf(mean,weight,n,ruser(sx:ex),ldx,m,isx,ip,ruser(sy:ey),wt, &
            rss,idf,b,se,cov,res,h,q,ldq,svd,irank,p,tol,wk,ifail)

!         Return the score (the residual sums of squares)
          score(i) = rss
        End Do

!       Keep track of the number of subsets evaluated
        iuser(5) = iuser(5) + max(1,la)
      End Subroutine f

      Subroutine prepare_user_arrays(m,iuser,ruser)
!       Populate the user arrays

!       M is the maximum number of features (as per H05ABF)

!       In this example RUSER holds the data required for the linear
!       regression (the matrix of explanatory variables, X, the vector of
!       response values, Y, and the tolerance, TOL) and IUSER holds number
!       of observations and the location of the various elements in RUSER

!       .. Scalar Arguments ..
        Integer, Intent (In)           :: m
!       .. Array Arguments ..
        Real (Kind=nag_wp), Allocatable, Intent (Out) :: ruser(:)
        Integer, Allocatable, Intent (Out) :: iuser(:)
!       .. Local Scalars ..
        Integer                        :: i, ierr, j, liuser, lruser, n, p1,   &
                                          p2
        Character (200)                :: line
!       .. Executable Statements ..
        Continue

!       Read in the number of observations for the data used in the linear
!       regression skipping any headings or blank lines
        Do
```

```
          Read (nin,*,Iostat=ierr) line
          If (ierr/=0) Then
            Exit
          End If
          Read (line,*,Iostat=ierr) n
          If (ierr==0) Then
            Exit
          End If
        End Do

!       Allocate space for the user arrays
        liuser = 5
        lruser = n + n*m + 1
        Allocate (ruser(lruser),iuser(liuser))

!       Number of observations
        iuser(1) = n
!       Location of TOL in RUSER
        iuser(2) = 1
!       Start of Y in RUSER
        iuser(3) = 2
!       Start of X in RUSER
        iuser(4) = iuser(3) + n
!       Keep track of the number of subsets evaluated
        iuser(5) = 0

!       Read in the tolerance for the regression
        Read (nin,*) ruser(iuser(2))

!       Read in the data
        p1 = iuser(3)
        p2 = iuser(4)
        Do i = 0, n - 1
          Read (nin,*)(ruser(p2+i+j*n),j=0,m-1), ruser(p1+i)
        End Do
      End Subroutine prepare_user_arrays
    End Module h05abfe_mod

    Program h05abfe

!     .. Use Statements ..
      Use nag_library, Only: h05abf, nag_wp
      Use h05abfe_mod, Only: f, nin, nout, prepare_user_arrays
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: gamma
      Integer                          :: i, ifail, ip, j, la, m, mincnt,      &
                                          mincr, mip, nbest
!     .. Local Arrays ..
      Real (Kind=nag_wp)               :: acc(2)
      Real (Kind=nag_wp), Allocatable  :: bscore(:), ruser(:)
      Integer, Allocatable             :: a(:), bz(:,:), ibz(:), iuser(:),     &
                                          z(:)
      Logical, Allocatable             :: mask(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: max, pack
!     .. Executable Statements ..
      Write (nout,*) 'H05ABF Example Program Results'
      Write (nout,*)

!     Skip headings in data file
      Read (nin,*)
      Read (nin,*)

!     Read in the problem size
      Read (nin,*) m, ip, nbest

!     Read in the control parameters for the subset selection
      Read (nin,*) mincr, mincnt, gamma, acc(1:2)
```

```
!     Allocate memory required by the subset selection routine
      mip = m - ip
      Allocate (z(mip),a(max(nbest,m)),bz(mip,nbest),bscore(max(nbest,m)))

!     Prepare the user workspace arrays IUSER and RUSER
      Call prepare_user_arrays(m,iuser,ruser)

!     Call the forward communication best subset routine
      ifail = -1
      Call h05abf(mincr,m,ip,nbest,la,bscore,bz,f,mincnt,gamma,acc,iuser,      &
        ruser,ifail)
      If (ifail/=0 .And. ifail/=42) Then
!       An error occurred
        Go To 100
      End If

!     Titles
      Write (nout,99999) '    Score        Feature Subset'
      Write (nout,99999) '    -----        --------------'

!     Display the best subsets and corresponding scores. H05AAF returns a list
!     of features excluded from the best subsets, so this is inverted to give
!     the set of features included in each subset
      Allocate (ibz(m),mask(m))
      ibz(1:m) = (/(i,i=1,m)/)
      Do i = 1, la
        mask(1:m) = .True.
        Do j = 1, mip
          mask(bz(j,i)) = .False.
        End Do
        Write (nout,99998) bscore(i), pack(ibz,mask)
      End Do

      Write (nout,*)
      If (ifail==42) Then
        Write (nout,99997) nbest,                                              &
          ' subsets of the requested size do not exist, only ', la,            &
          ' are displayed.'
      End If
      Write (nout,99996) iuser(5), ' subsets evaluated in total'

100   Continue

99999 Format (1X,A)
99998 Format (1X,E12.5,100(1X,I5))
99997 Format (1X,I0,A,I0,A)
99996 Format (1X,I0,A)

    End Program h05abfe
```

10.2 Program Data

```
H05ABF Example Program Data
Data required by H05ABFE
14 5 3                      :: M,IP,NBEST
0 -1 -1.0 -1.0 -1.0         :: MINCR, MINCNT, GAMMA, ACC(1:2)

Data required by the scoring function
40                          :: N
1e-6                        :: TOL
 -1.59   0.19   0.40   0.43  -0.40   0.79   0.06
  0.33   1.60   0.58  -1.12   1.23   1.07  -0.07          -2.44
 -0.25   0.61  -0.36   1.16   0.61  -2.05  -0.02
 -0.04   0.80  -0.73  -0.63  -0.75  -0.73   1.43          -2.97
 -2.28   0.46  -0.65   0.33   0.16  -0.21  -1.61
 -0.54   0.48   0.37  -0.95  -2.14   0.48   2.02           7.42
 -0.52   1.05   0.64   0.02  -1.12   0.23   0.06
 -1.26   1.40  -0.98   2.47   0.49  -0.02  -0.05           3.00
 -0.84   1.86   0.10   0.73  -1.41   0.98   0.20
 -0.89   1.84   2.56   0.60  -0.12   0.71   0.23           8.83
```

```
 1.12  -0.51  -0.58   0.09  -1.14   2.11  -0.11
-0.34  -1.04  -0.43  -0.01  -0.38   1.80   0.05          0.03
 0.06   0.85  -2.09   0.22  -1.35  -0.36   1.20
 0.41   0.80  -0.28   0.18   0.27   0.92   0.63          2.57
-0.48  -1.02   0.08  -0.06   0.13  -1.18   2.30
 0.03   0.45   0.62  -1.97   0.97   0.93  -0.18          8.31
 0.08  -0.31   0.43  -0.38   0.01   1.30   0.66
 0.65  -0.59   0.76   0.04   0.17  -0.76  -0.90          4.55
 0.66   1.14   0.40   2.37   1.10   0.17  -0.38
 1.15  -1.00  -0.13  -0.69  -0.62  -0.18   0.00        -23.10
-1.08  -0.21  -1.13  -0.79  -0.76  -1.58   0.38
-0.03   1.26  -0.51  -0.75   0.86   0.29   0.68          3.38
-0.74  -1.59  -0.58  -1.09   1.18  -1.70  -1.02
 0.36   1.05   1.30  -0.98  -1.36  -1.28  -1.32         -0.13
 0.40  -1.58  -1.30  -0.10  -1.34   0.65  -0.56
 0.39  -0.73  -0.32   2.19  -0.49   0.69   0.18          5.47
 0.75  -3.09  -0.61  -1.89   0.15   0.77  -0.49
-0.63   1.20  -0.04   1.02   0.31   0.81  -0.45         13.97
-0.65   1.57  -1.50  -1.45   0.21   0.06   0.24
 2.24  -1.34   0.30   1.39  -0.38  -0.71   0.48         20.94
 1.36   1.40  -1.40  -0.90   0.36  -0.21  -0.97
 0.36  -0.26   0.08   0.06  -1.49   0.43  -1.61        -12.87
-1.01   1.50  -0.61  -0.25  -1.01  -0.43   1.90
-1.33  -0.96  -0.02   0.51  -1.38  -0.78   1.82         28.26
 1.34   1.02   3.50   0.10   0.50   0.04   0.61
-0.57  -2.69  -0.64  -0.34  -0.21  -1.97  -0.19          6.89
 0.29   0.67  -0.38  -0.63  -0.24   1.21  -0.09
 0.90  -2.20   1.72   0.29   0.66   0.19  -0.57          5.37
 0.67  -0.56  -0.41   1.22  -0.30   0.77   0.82
 0.36   1.18   1.87  -1.48   0.52   1.35   0.13         -1.50
-0.40  -1.10  -0.83   0.71   1.99  -0.24   1.30
-0.34  -0.70   0.28   0.16   0.27   0.37  -1.79        -23.00
-0.78   0.60  -0.45  -0.26  -0.23   0.89   0.87
 1.01   1.20   0.28   0.79   2.76   0.35   1.31         14.09
-1.29   0.62  -0.59   1.52   0.62   0.21   1.31
 1.09  -0.36  -0.34  -0.03  -0.59  -1.70  -0.03        -11.05
 0.40  -1.45  -0.98   2.10  -1.09  -0.53  -0.38
-1.36   0.13   0.70  -1.51   0.08  -0.62  -0.64        -32.04
 0.43  -0.86   0.70  -1.07  -0.76   0.72  -0.14
-1.58   0.00   0.58  -0.21   1.30   2.02   1.52         23.36
-0.48   0.01   1.30   0.58  -0.54   1.09   0.91
 2.90   1.32  -1.20  -0.59  -0.51   0.20  -1.74         -5.58
-1.32  -1.41  -0.58  -1.29   1.61  -0.35  -0.72
-1.92  -1.09   0.56  -0.87  -0.71   1.25   0.10          2.48
 1.43   0.69   1.34  -0.32   2.84  -1.43  -0.47
-0.01   0.83  -0.72  -0.78   0.50  -1.22   0.54         -5.30
 0.82   0.46   0.15  -0.57   0.93   1.33  -0.23
-1.07   0.76   0.25  -1.96   0.39   0.24  -0.26         -7.77
-0.91   0.23  -0.19   1.58  -0.27   0.33  -0.60
-1.39  -0.30  -0.81  -0.95   0.88  -0.09  -0.35        -34.25
 0.65  -1.14   1.18  -1.06  -0.68  -0.22   0.21
 0.94   1.08   0.81  -0.33   0.42  -0.90   0.49         26.78
-0.36  -0.50  -0.02  -0.04   0.77   0.62  -1.35
-0.64   1.20   1.22   0.18  -1.39  -0.81  -0.99        -11.85
-1.82   1.06   0.28   0.14   0.62  -0.80  -1.08
-2.15   1.37   1.57  -1.48  -0.79   0.28  -0.20         -8.62
 1.54   0.50   0.13  -0.68   0.26  -1.13   0.62
-0.43   0.39   1.14   0.15   1.03   0.46   0.40         12.35
-1.61  -0.61   0.93  -0.37   0.44  -1.45   0.58
-1.77   0.72  -2.05  -0.03  -1.24  -1.40  -0.06         -1.54
-0.48   0.67   0.04   0.27  -0.84  -0.06  -3.67
 0.09   1.66  -0.30   1.67   1.08   0.00   0.43        -16.59
-1.65  -1.16  -1.17   1.12   0.11  -0.15   0.48
-1.72   1.08  -0.94   0.49  -0.56   0.95   1.09         -8.69
-0.85  -0.02   1.18  -1.16   0.49   1.56  -0.60
 0.32   0.72  -1.20   2.52   1.78   0.16  -0.01          7.82
-0.60  -0.73  -1.23   1.50   0.40  -0.20  -0.65
 0.68   1.09   0.40  -1.50  -2.10   0.21  -0.18        -18.56
-0.66  -0.01  -0.01   0.85  -2.04   1.17  -0.56
 1.72  -0.18   1.14  -0.96  -0.92  -0.28   1.58         17.21 :: X,Y
```

10.3 Program Results

```
 H05ABF Example Program Results

     Score        Feature Subset
     -----        --------------
  0.10475E+04     4     7     8    10    14
  0.10599E+04     4     5     7     8    14
  0.10702E+04     4     5     7    10    14

 45 subsets evaluated in total
```