Appendix
MCDA Tools and Meta-Heuristic Techniques: Sample Codes

Computer Codes

A.1 MATLAB code for Analytic Hierarchy Process (AHP)

```
**********************************************************************
                    This code is developed on the algorithm proposed by T.L.Saaty for AHP

**********************************************************************
tic;
clc;
clear % is must required to clean workspace data %
row = input('Enter no of rows or columns :');
for i = 1:1:row
  for j= 1:1:row
    if i==j
     a(i,j)=1;
        elseif (i>j)
                a(i,j)= 1/a(j,i) ;
    else
       a(i,j)=input('Enter pair wise comparison value :');

    end;

  end
  end
disp('Pair Wise Comparison Matrix')
disp(a)

**********************************************************
% Normaliztion of pair wise matrix %

for i=1:1:row
    Sum_Col=0.0;
    for j=1:1:row
       Sum_Col= Sum_Col + a(j,i);
    end
       Col_Sum(i)= Sum_Col;
end
disp(Col_Sum)
for i=1:1:row
    for j=1:1:row
       norm_mat(j,i)= a(j,i)/Col_Sum(i);
    end
end
```


K. Mukherjee, *Supplier Selection*, Studies in Systems, Decision and Control 88,
DOI 10.1007/978-81-322-3700-6

```
disp('Normalized Matrix')
disp(norm_mat)
% Take average of row to calculate priority %
for i=1:1:row
   row_sum = 0.0;
      for j=1:1:row
            row_sum = row_sum + norm_mat(i,j);
      end
   prio_mat(i)= row_sum/row;
end
disp('Priority Vectors')
disp(prio_mat)
************************************************
% Calculation of consistency index %

lambda_max=0.0;
for i=1:1:row
   lambda_max= lambda_max + Col_Sum(i)* prio_mat(i);
end
CI = (lambda_max - row) /(row -1);
disp('Consistency Index')
disp(CI)
% Random Consistency Index Formula%
RI=1.98*(row-2)/row;
disp('Random Consistency Index')
disp(RI);
CR=CI/RI;
disp('Consistency Ratio')
disp(CR)

************************************************
%Pair Wise Comparison Matrix for Alternatives%

Alt_row =input('Enter No. of Alternatives:');
for k=1:1:row  %Comparison with respect to each criterion%
   disp(['Comparison with respect to Criteria :  ', num2str(k)])
   for i=1:1:Alt_row
   for j=1:1:Alt_row
  if i==j
    alt(i,j,k)=1;
       elseif (i>j)
              alt(i,j,k)= 1/alt(j,i,k) ; % Array of Matrix representation %
   else
      alt(i,j,k)=input('Enter pair wise comparison value :');

  end;
    end
    end
end
for i=1:1:row
   disp(alt(:,:,i))
end

*********************************************************
% Normaliztion of pair wise matrix %
```

```
for k=1:1:row
for i=1:1:Alt_row
   Sum_Col=0.0;
   for j=1:1:Alt_row
      Sum_Col= Sum_Col + alt(j,i,k);
   end
      Col_Sum_alt(k,i)= Sum_Col;
end
end

for i=1:1:row
disp(Col_Sum_alt(i,:))

end

for k=1:1:row
for i=1:1:Alt_row
   for j=1:1:Alt_row
      norm_mat_alt(j,i,k)= alt(j,i,k)/Col_Sum_alt(k,i);
   end
end
end
disp('Normalized Alternative Matrix')
for i=1:1:row
   disp(norm_mat_alt(:,:,i))
end

************************************************
 % Take average of row to calculate priority %

for k=1:1:row
for i=1:1:Alt_row
   row_sum = 0.0;
       for j=1:1:Alt_row
            row_sum = row_sum + norm_mat_alt(i,j,k);
       end
   prio_mat_alt(k,i)= row_sum/Alt_row;
end
end

disp('Local Priority Vectors Of Alternative')
for i=1:1:row
  disp(prio_mat_alt(i,:))
end

************************************************
% Calculation of CR of each pair wise comparison matrix of alternatives %

for i=1:1:row
   lambda_max_alt=0.0;
   for j=1:1:Alt_row
   lambda_max_alt= lambda_max_alt + Col_Sum_alt(i,j)* prio_mat_alt(i,j);
   end
   lambda_max_new(i)= lambda_max_alt;
end
for i=1:1:row
CI_alt(i) = (lambda_max_new(i) - Alt_row) /(Alt_row -1);
end
```

```
disp('Consistency Index of Pairwise Comparion Matrix of Alternatives')
disp(CI_alt)

****************************************
% Random Consistency Index Formula%

RI=1.98*(Alt_row-2)/Alt_row;
disp('Random Consistency Index')
disp(RI);
for i=1:1:row
CR_alt(i)=CI_alt(i)/RI;
end
disp('Consistency Ratio of Pairwise Comparison Matrix of Alternatives')

 disp(CR_alt)

*********************************************
 %disp('Global Priority Vectors Of Alternative')

for i=1:1:Alt_row
   Prio_Sum =0.0;
  for j=1:1:row
   Prio_Sum = Prio_Sum + prio_mat(j)* prio_mat_alt(j,i);
  end
  global_prio(i)=Prio_Sum;
end
disp('Rank of Alternativies:')
sort_global_prio = sortrows(global_prio);
for i=1:1:Alt_row
       for j=1:1:Alt_row
if sort_global_prio(i)==global_prio(j)
   rank(j)= i;
end
    end
end
for i=1:1:Alt_row
disp(['Alternative :  ', num2str(i),'  Global Priority:  ',num2str(global_prio(i)), '   Rank  :  ',num2str(rank(i))])
end
toc;
```

A.2 MATLAB Code for Fuzzy AHP by alpha-cut method

```
*************************************************************************
                                This code is developed on fuzzy AHP alpha–cut method

*************************************************************************
tic;
clc;
clear % is must required to clean workspace data %
row = input('Enter no of rows or columns :');
for i = 1:1:row
  for j= 1:1:row
    if i==j
     a(i,j)=1;
        elseif (i>j)
                a(i,j)= 1/a(j,i) ;
```

```
    else
       a(i,j)= Fuzzy_Alpha_Cut();

    end;

  end
  end
disp('Pair Wise Comparison Matrix')
disp(a)

*****************************************
 % Normaliztion of pair wise matrix %

for i=1:1:row
    Sum_Col=0.0;
    for j=1:1:row

        Sum_Col= Sum_Col + a(j,i);
    end
        Col_Sum(i)= Sum_Col;
end
disp(Col_Sum)
for i=1:1:row
    for j=1:1:row
        norm_mat(j,i)= a(j,i)/Col_Sum(i);
    end
end
disp('Normalized Matrix')
disp(norm_mat)

***********************************************
% Take average of row to calculate priority %

for i=1:1:row
    row_sum = 0.0;
         for j=1:1:row
                row_sum = row_sum + norm_mat(i,j);
         end
    prio_mat(i)= row_sum/row;
end
disp('Priority Vectors')
disp(prio_mat)

***********************************************
% Calculation of consistency index %

lambda_max=0.0;
for i=1:1:row
    lambda_max= lambda_max + Col_Sum(i)* prio_mat(i);
end
CI = (lambda_max - row) /(row -1);
disp('Consistency Index')
disp(CI)

***********************************************
% Random Consistency Index Formula%
RI=1.98*(row-2)/row;
disp('Random Consistency Index')
```

```
disp(RI);
CR=CI/RI;
disp('Consistency Ratio')
disp(CR)

**************************************************
%Pair Wise Comparison Matrix for Alternatives%

Alt_row =input('Enter No. of Alternatives:');
for k=1:1:row  %Comparison with respect to each criterion%
   disp(['Comparison with respect to Criteria :  ', num2str(k)])
   for i=1:1:Alt_row
   for j=1:1:Alt_row
 if i==j
    alt(i,j,k)=1;
       elseif (i>j)
              alt(i,j,k)= 1/alt(j,i,k) ; % Array of Matrix representation %

   else
      alt(i,j,k)= Fuzzy_Alpha_Cut();

 end;
   end
   end
end
for i=1:1:row
   disp(alt(:,:,i))
end

**************************************************
% Normaliztion of pair wise matrix %

for k=1:1:row
for i=1:1:Alt_row
   Sum_Col=0.0;
   for j=1:1:Alt_row
      Sum_Col= Sum_Col + alt(j,i,k);
   end
      Col_Sum_alt(k,i)= Sum_Col;
end
end

for i=1:1:row
disp(Col_Sum_alt(i,:))
end

for k=1:1:row
for i=1:1:Alt_row
   for j=1:1:Alt_row
      norm_mat_alt(j,i,k)= alt(j,i,k)/Col_Sum_alt(k,i);
   end
end
end
disp('Normalized Alternative Matrix')
for i=1:1:row
   disp(norm_mat_alt(:,:,i))
end
```

```
**************************************************
 % Take average of row to calculate priority %

for k=1:1:row
for i=1:1:Alt_row
   row_sum = 0.0;
       for j=1:1:Alt_row
             row_sum = row_sum + norm_mat_alt(i,j,k);
       end
   prio_mat_alt(k,i)= row_sum/Alt_row;
end
end

disp('Local Priority Vectors Of Alternative')
for i=1:1:row
  disp(prio_mat_alt(i,:))
end
**********************************************************
% Calculation of CR of each pair wise comparison matrix of alternatives %

for i=1:1:row
   lambda_max_alt=0.0;
   for j=1:1:Alt_row
   lambda_max_alt= lambda_max_alt + Col_Sum_alt(i,j)* prio_mat_alt(i,j);
   end
   lambda_max_new(i)= lambda_max_alt;
end
for i=1:1:row
CI_alt(i) = (lambda_max_new(i) - Alt_row) /(Alt_row -1);
end
disp('Consistency Index of Pairwise Comparion Matrix of Alternatives')
disp(CI_alt)

**************************************************
% Random Consistency Index Formula%

RI=1.98*(Alt_row-2)/Alt_row;
disp('Random Consistency Index')
disp(RI);
for i=1:1:row
CR_alt(i)=CI_alt(i)/RI;
end
disp('Consistency Ratio of Pairwise Comparison Matrix of Alternatives')
disp(CR_alt)

**********************************************************
%disp('Global Priority Vectors Of Alternative')

for i=1:1:Alt_row
   Prio_Sum =0.0;
  for j=1:1:row
    Prio_Sum = Prio_Sum + prio_mat(j)* prio_mat_alt(j,i);
  end
  global_prio(i)=Prio_Sum;
end
disp('Rank of Alternatives:')
sort_global_prio = sortrows(global_prio);
```

```
for i=1:1:Alt_row
      for j=1:1:Alt_row
if sort_global_prio(i)==global_prio(j)
   rank(j)= i;
end
   end
end
for i=1:1:Alt_row
disp(['Alternative :  ', num2str(i),'  Global Priority:  ',num2str(global_prio(i)), '   Rank  :  ',num2str(rank(i))])
end
toc;
```

A.3 MATLAB Code for Extent Fuzzy AHP

```
**************************************************************************
                    This code is developed on algorithm proposed by D.Y.Chang for Extent Fuzzy AHP

**************************************************************************
tic;
clc;
clear % is must required to clean workspace data %
row = input('Enter no of rows or columns :');
for i = 1:1:row
  for j= 1:1:row

   if i==j
    a(i,j,1)=1;
    a(i,j,2)=1;
    a(i,j,3)=1;
       elseif (i>j)
          for k=1:1:3
            n=4-k;
              a(i,j,k)= 1/a(j,i,n) ;
              end;
   else
      a1= Extent_Fuzzy_AHP_Linguistic();
      a(i,j,1)=a1(1);
       a(i,j,2)=a1(2);
      a(i,j,3)=a1(3);
    end
  end
  end
sum_u=0.0;
sum_m=0.0;
sum_l=0.0;
for i=1:1:row
   for j=1:1:row
         sum_u=sum_u+a(i,j,3);
         sum_m=sum_m+a(i,j,2);
         sum_l=sum_l+a(i,j,1);

   end
end
disp('Sum of L M U:')
disp(sum_l)
disp(sum_m)
disp(sum_u)
```

```
**************************************************
%Creation of Cell Array %

C=cell(row,3);
for i=1:1:row
   sum_u1=0.0;
   sum_m1=0.0;
   sum_l1=0.0;
   for j=1:1:row
         sum_u1=sum_u1+ a(i,j,3);
         sum_m1=sum_m1+a(i,j,2);
         sum_l1=sum_l1+a(i,j,1);

   end
   sum_l2(i)=sum_l1;
   sum_m2(i)=sum_m1;
   sum_u2(i)=sum_u1;
   C(i,:)={sum_l2(i) sum_m2(i) sum_u2(i)};

end
disp('Extent Fuzzy Synthetic Value')
disp(C)

**************************************************
% Calculation of fuzzy synthetic extent value%

for i=1:1:row
          S(i,1)= sum_l2(i)/sum_u;
          S(i,2)=sum_m2(i)/sum_m;
          S(i,3)=sum_u2(i)/sum_l;
end
disp(S)

**************************************************
% Calculation of Degree Possibility%

for j=1:1:row
   for i=1:1:row
  if i==j
     Val(i)=196;
  else
  Val(i)= Degree_Possibility(S(j,1),S(j,2),S(j,3),S(i,1),S(i,2),S(i,3));
  end
  end
min_pos(j) = min(Val);
end
% Calculate Priority Vector%
disp('Local Priority Vector:')
disp(min_pos)

**************************************************
%Calculate Normalized Priorirty Vector%

norm_sum=0.0;
for i=1:1:row
 norm_sum=norm_sum+min_pos(i);
end
for i=1:1:row
```

```
    norm_priority(i) = min_pos(i)/norm_sum;
end
disp('Normalized Priority')
disp(norm_priority)
toc;
```

A.4 MATLAB Code for Degree of possibility

```
function Poss = Degree_Possibility(l2,m2,u2,l1,m1,u1)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
if m2>=m1
    Poss=1;
elseif l1>=u2
    Poss=0;
else
    deno=(m2-u2)-(m1-l1);
    Poss= (l1-u2)/deno;
end
```

A.5 MATLAB Code for Extent Fuzzy AHP(linguistic variables)

```
function fuzzy_val  = Extent_Fuzzy_AHP_Linguistic()
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
% Declaration of TFN
Y=input('Enter Yes or No if you want to flip comparison:','s');
S = input('Enter any of the following linguistic terms : absolute,very strong,fairly strong,weak,equal:','s');
if   strcmp(Y,'No')==1
switch lower(S)
   case 'absolute'
      fuzzy_val = [3.5;4;4.5];
   case 'very strong'
      fuzzy_val = [2.5;3;3.5];
   case 'fairly strong'
      fuzzy_val = [1.5;2;2.5];
   case 'weak'
      fuzzy_val = [.67;1;1.5];
    case 'equal'
      fuzzy_val = [1;1;1];
   otherwise
      disp('Unknown Linguistic Term.')
end
else
    switch lower(S)
   case 'absolute'
       fuzzy_val = [0.2222;0.25;0.2857];
   case 'very strong'
      fuzzy_val = [0.2857;0.3333;0.4];
   case 'fairly strong'
      fuzzy_val = [0.4;0.5;0.6666];
   case 'weak'
      fuzzy_val = [0.6666;1;1.4925];

    case 'equal'
      fuzzy_val = [1;1;1];
   otherwise
```

```
    disp('Unknown Linguistic Term.')
  end
end
```

A.6 MATLAB code for Fuzzy Alpha Cut Method

```
*************************************************************************
                    This computer code is developed for fuzzy alpha-cut method
*************************************************************************

function a_alpha_beta = Fuzzy_Alpha_Cut()
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
% Declaration of TFN
S = input('Enter any of the following linguistic terms : absolute,very strong,fairly strong,weak,equal:','s');
switch lower(S)
  case 'absolute'
    fuzzy_val = [3.5;4;4.5];
  case 'very strong'
    fuzzy_val = [2.5;3;3.5];
  case 'fairly strong'
    fuzzy_val = [1.5;2;2.5];
  case 'weak'
    fuzzy_val = [.67;1;1.5];
   case 'equal'
    fuzzy_val = [1;1;1];
  otherwise
    disp('Unknown Linguistic Term.')
end
% Defuzzification of fuzzy value by fuzzy alpha cut %
%alpha = input('Enter fuzzy alpha-cut value : ');
%beta=input('Enter degree of satisfaction value : ');
alpha=0.5;
disp(fuzzy_val(2))
a_alpha_lower = (fuzzy_val(2)-fuzzy_val(1))* alpha + fuzzy_val(1);
a_alpha_upper = fuzzy_val(3)-(fuzzy_val(2) -fuzzy_val(3))* alpha;
%a_alpha_beta = (1-beta)* a_alpha_lower + beta * a_alpha_upper;

a_alpha_beta (1) =(a_alpha_lower);
a_alpha_beta (2)=(a_alpha_upper);
```

A.7 MATLAB Code for Order Allocation to selected suppliers by genetic algorithm-I

```
*************************************************************************

Allocation of order to selected supplier/s – An example of single objective constrained optimization

*************************************************************************

% GA FITNESS FUNCTION %
```

```
function y = simple_fitness_supplier(x)
y=-.30561*x(1)-.38463*x(2)-.30977*x(3);

**************************************************************************

% CONSTRAINT OPTIMIZATION %
function [c,ceq]=constraint_supplier(x)
c=[30*x(1)+60*x(2)+35*x(3)-65000;0.01*x(1)+0.02*x(2)+0.04*x(3)-48];
ceq=x(1)+x(2)+x(3)-1200;

**************************************************************************
clc;
objectiveFunction=@simple_fitness_supplier;
nvars=3;
LB=[0 0 0];
UB=[650 650 550];
constraintFunction=@constraint_supplier;

options=gaoptimset('PopulationSize',20,'CrossoverFraction',0.8,'MutationFcn',{@mutationadaptfeasible,0.05});
options=gaoptimset(options,'PlotFcns',{@gaplotbestf,@gaplotdistance, @gaplotrange,@gaplotbestindiv
    },'Display','iter','Generations',60);
[x,fval]=ga(objectiveFunction,nvars,[ ],[ ],[ ],[ ],LB,UB,constraintFunction,options)
```

A.8 MATLAB Code for Order Allocation to selected suppliers by genetic algorithm-II

```
**************************************************************************
Order allocation to selected supplier/s – An example of multi-objective genetic algorithm (MOGA)
**************************************************************************
function y = supplier_selection_multiobjective_fitness(x)
y(1)= 3859 * x(1)+ 3850 * x(2)+ 3851 * x(3);
y(2)= -.5654 * x(1)-.5024 * x(2)-.2033 * x(3) ;
y(3)= 0.1 * x(1) + 0.15 * x(2) + 0.2 * x(3);
y(4)= 0.2 * x(1) + 0.25 * x(3) + 0.3 * x(3);
y(5)=0.15 * x(1) + 0.2 * x(2) + 0.2 * x(3);

**************************************************************************
clc;
tic;
FitnessFunction=@supplier_selection_multiobjective_fitness;
numberOfVariables=3;
A=[2760 2750 2749]; b=[28000000];
Aeq=[1 1 1];beq=[9900];
lb=[0 0 0];
ub=[4000 3000 3000];

options=gaoptimset('PlotFcns',{@gaplotpareto});
options=gaoptimset(options,'PopulationSize',80,'HybridFcn',[],'CrossoverFraction',0.85,'CrossoverFcn',@crossoverarithmetic,'M
utationFcn',{@mutationadaptfeasible,0.5});
options = gaoptimset(options,'DistanceMeasureFcn',{@distancecrowding,'genotype'});
options = gaoptimset(options,'ParetoFraction',0.5,'Display','iter');
%options = gaoptimset(options,'PopulationSize',20);
[x,fval,exitFlag,Output,population,scores]=gamultiobj(FitnessFunction,numberOfVariables,A,b,Aeq,beq,lb,ub,options);
display(scores)
display(population)
display(fval)
fprintf('The number of points on the Pareto front was: %d\n', size(x,1));
fprintf('The average distance measure of the solutions on the Pareto front was: %g\n', Output.averagedistance);
fprintf('The spread measure of the Pareto front was: %g\n', Output.spread);
disp(['Elapsed time to solve multi-objective GA is ', num2str(toc)]);
```

A.9 MATLAB Code for Singular Value Decomposition method for fuzzy rule base reduction method

```
**************************************************************************
                This computer code is developed on SVD method to reduce fuzzy rule base
**************************************************************************
function Z = Single_Value_Decomposition()
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
clc
A=[1.2857 1.2857 4 4;1.2857 4 4 6;4 4 6 6;4 6 6 8.7143] %stage-1 sustainable
%supplier selection
%A=[-10 -7.5 -5 -2.5 0;-7.5 -5 -2.5 0 2.5;-5 -2.5 0 2.5 5;-2.5 0 2.5 5 7.5;0 2.5 5 7.5 10]
%A=a;
%A=[0.8 0.8 2.5 4.5;0.8 2.5 4.5 5.5;2.5 4.5 5.5 7.5;4.5 5.5 7.5 9.2]
rank_A=rank(A);
disp(rank_A)
[U S V]=svd(A);
disp('U matrix')
disp(U)
disp('V matrix')
disp(V)
disp('Diagonal Matrix of Singular Value')
disp(S)
disp('Rank of Diagonal Matrix of Singular Value')
K=rank(S);
disp(K)
if (K>=2)
    K=2;
    disp('2 input is considered')
end
M=size(U);
N=M(2);
B=size(U);
l=B(1);
UR=U(:,1:K);
UD=U(:,K+1:N);
Cu=blkdiag(sum(UR(:,1)),sum(UR(:,2)),1);%To form the diagonal matrix
disp(Cu)
UR(1,K+1)=0;
U1= UR*Cu;
min_U1=sort(U1);
Col_U1=size(U1);

if (min_U1>= -1)
    delta=1;

else
    delta= 1/mod(min_U1);

end

for i=1:1:K
  for j=1:1:K

      if (i==j)
```

```
        stoch_matrix(i,j)=1+delta;
      else
      stoch_matrix(i,j)=1;
      end
    end

end

final_stoch_matrix =(1/(2+delta))*stoch_matrix;
U1_temp=U1(:,1:K);
U2=U1_temp*final_stoch_matrix;
disp('Prototypical Value')
disp(U2)
B=size(U2);
M=B(1);
************************************************
 %Prototypical Membership Value for FIRST Input%

EU=[U2(1,:);U2(M,:)];
%EU(K,1)=0;
EU_inv=inv(EU);
disp(EU_inv)
U3=U2*EU_inv;
disp('Membership Value for first input:')
disp(U3)


*************************************************
 %Overlapping Membership Function: Matrix V%

%V=V';
VR=V(:,1:K);
MV=size(V);
NV=MV(2);
VD=V(:,K+1:NV);
Cv=blkdiag(sum(VR(:,1)),sum(VR(:,2)),1);%To form the diagonal matrix
VR(1,3)=0;
V1= VR*Cv;
min_V1=sort(V1);
if (min_V1>= -1)
   deltaV=1;

else
   deltaV= 1/mod(min_V1);

end

for i=1:1:2
  for j=1:1:2

      if (i==j)
         stoch_matrixV(i,j)=1+deltaV;
      else
      stoch_matrixV(i,j)=1;
      end
    end

end
```

```
final_stoch_matrixV =(1/(2+delta))*stoch_matrixV;
V1_temp=V1(:,1:2);
V2=V1_temp*final_stoch_matrixV;

*****************************************************
%Prototypical Membership Value: FOR SECOND INPUT%

BV=size(V2);
V=BV(1);
EV=[V2(1,:);V2(V,:)];
EV_inv=inv(EV);
V3=V2*EV_inv;
disp('Membership Value for second input:')
disp(V3)

*****************************************************
% Reduced matrix of rule consequent values%

disp('Inverse of Eu')
inv(EU_inv)
disp('Inverse of Du')
inv(final_stoch_matrix)
disp('Inverse of Cu')
Cu1=Cu(1:2,1:2);
inv(Cu1)
S(1:2,1:2)
disp('Inverse of Cv')
Cv1=Cv(1:2,1:2);
inv(Cv1)
disp('Inverse of Dv')
inv(final_stoch_matrixV)
disp('Inverse of Ev')
inv(EV_inv')
Z=inv(EU_inv)*inv(final_stoch_matrix)*inv(Cu1)*S(1:2,1:2)*inv(Cv1)*inv(final_stoch_matrixV)*inv(EV_inv');

end
```

A.10 VB.NET Code for database connection used to prepare DSS

```
*****************************************************
A simple VB.NET code to connect SQL SERVER with front end with ADO
*****************************************************
Imports System.Data
Imports System.Data.SqlClient
Imports System.IO
Imports System.Data.Common
Partial Class _Default
    Inherits System.Web.UI.Page
    Private connect As String
    Private price(1000) As Double
    Private temp1(100) As Double
    Dim _count As Integer

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
        Dim cl As New Class_connect
        connect = cl.init_db_con
        If Page.IsPostBack = False Then
            Button1.Text = "Load Data"
        End If
        If Page.IsPostBack = True Then
            Button1.Text = "Save"
```

```
        Button1.Enabled = False
        Dim con As New SqlConnection(connect)
        con.Open()
        Dim sql As String
        sql = "Select Name,Prod_Id,Prod_Des,Price,Quantity from Supplier where Prod_Id=1002"
        'Paramteric SQL connection
        Dim myadap As New SqlDataAdapter(sql, con)
        Dim myds As New DataSet()
        myadap.Fill(myds, "Orders")
        con.Close()
        'Only the name of the field of database is required
        DDgrdVw.DataSource = myds.Tables("Orders")
        'DDgrdVw.DataValueField = "Prod_Id"
        DDgrdVw.DataBind()

        DDgrdVw.SelectedIndex = -1
        Dim i As Integer
        'Dim j As Integer
        Dim _maxValue As Double
        Dim sum As Double
        _count = myds.Tables("Orders").Rows.Count - 1
        Dim temp(_count) As Double
        'Dim temp1(j) As Double
        ReDim temp1(_count)
        ReDim price(_count)
        For i = 0 To myds.Tables("Orders").Rows.Count - 1
            price(i) = CDbl(myds.Tables("Orders").Rows(i).Item("Price").ToString)
            temp(i) = price(i)
        Next

        temp1 = temp
        Array.Sort(temp1)
        _maxValue = temp1(_count).ToString
        sum = 0.0

        For i = 0 To _count
            temp1(i) = _maxValue - price(i)
            sum = sum + temp1(i)
        Next
        For i = 0 To _count
            temp1(i) = temp1(i) / sum

        Next

    End If

End Sub

Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim st As Stream = File.Open("C:\price.xls", FileMode.Create, FileAccess.Write)
    Dim bw As New StreamWriter(st)
    Dim i As Integer
    For i = 0 To _count
        Dim str1 As String
        str1 = CDbl(temp1(i).ToString)
        bw.Write(str1)
        bw.WriteLine()
    Next
    bw.Close()
End Sub
End Class
```