GMHS technical report

1- GMHS-benchMark pseudocode:

Input:

HMS, N (archiveSize), maxIter, HMCR, Pc, Pbw, bw1Min, Dim, LB, HB, alpha, initialPAR, phi, initialGama1, initialGama2

Pgm, sigma , nobj, psi

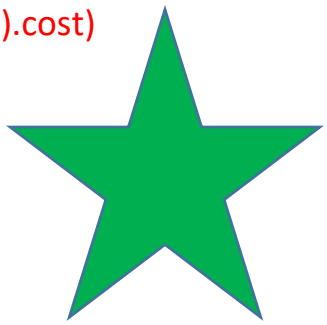
Output:

paretoSet

start:

```
1- Initial_HM=unifrnd(LB,HB,HMS,Dim)
2- HM= Initial_HM
3-      A=[] // archive
4- For i=1:HMS
       HM(I).cost=objfun(HM(i))
   End
5- HM = sort(HM) //fast non dominated sorting
6- For iter=1:maxIter
       For h=1:HMS
          Xnew=[]
          For j=1:Dim
              If rand<HMCR
                 If rand<Pc
                    randIdx=round(unifrnd(1,HMS))
                    Xnew(j)= HM(rndIdx,j)
                  Else
                     Xnew(j)=HM(h,j)
                  End
               Else
                  Xnew(j)=unifrnd(LB,HB)
              End
               If iter==1
                    prevPAR=initial_PAR
                end
```

```
PAR= psi*prevPAR*(1-prevPAR)
If rand<=PAR
    If rand<=Pbw
       Bw1Max(j)=(HB(j)-LB(j))/2*HMS
       Bw1(j)=bw1Min(j)+(bw1Max(j)-bw1Min(j))*((maxIter-
       iter)/maxIter)^phi
       If iter==1
            prevGama1=initialGama1
       end
       Gama1=sin(alpha/prevGama1)
       Xnew(j)=Xnew(j)+bw1(j)*Gama1
    Else
        If iter==1
             prevGama2=initialGama2
        end
        Gama2=sin(alpha/prevGama2)
        R1= round(unifrnd(1,HMS))
        R2= round(unifrnd(1,HMS))
        While r1==r2
             R2= round(unifrnd(1,HMS))
        End
        Mean=HM(r1,j)-HM(r2,j)
        Sigma=mean/10
        Bw2(j)=normrnd(mean,sigma)
        Xnew(j)=Xnew(j)+bw2(j)*gama2
     End
     Xnew=CheckBounds(Xnew,LB,HB)
End // end if PAR

Xnew=GaussianMutation(Xnew,Pgm,sigma)

Xnew.cost= objfun(Xnew)

If Dominates(Xnew.cost,HM(h).cost)

   HMnew(h)=Xnew

Else

   HMnew(h)= HMnew(h)
```



End

End // end for h=1:HMS

HM=[HM ; HMnew] // 2*HMS

HM=sort(HM) // fast non-dominated-sorting

HM= selectionOperator(HM) % using crowding distance as nsga2 // HMS

NonDominatedHarmony=findNonDominated(HM)

A=[A; HM(NonDominatedHarmony)]

M=size(A)

While M > N

If nobj==2

For a=1:M

A(a).dist=crowdingDist(A(a))

End

Idx= find(min(A.dist))

A(idx)=[]

Else

A= TrunctionProcedure(A)

End

End // end iterations

7- paretoSet=A

end

Gaussian Mutation:

Input:

Xnew,sigma

Output:

```
Xnew
Statrt:
For j=1:Dim
    If rand<Pgm
        Xnew(j)=normrnd(Xnew(j),sigma)
        Xnew(j)=checkBound(Xnew(j),LB,HB)
    Else
        Xnew(j)=Xnew(j)
    End
End
End
```

This pseudocode has introduced very bad results, but the reason is the following:

Going back to the red part of the code where the star is located, we will notice that the Xnew isn't moved to the new HM if it does not dominate the previous solution. And this section is not clearly illustrated in the paper, so we have confused about it.

---- but after more checking and reviewing, I thought that the paper's writer had not wanted to refuse the Xnew anyway ,and he inserts it into the new HM.

---- so I have replaced the red section of the pseudocode with the following:

```
newHM(h,:) = Xnew
```

---- although the simplicity of this tiny modification, but it leads to large improvement

So the results of GMHS-BenchMark have finally been very good, and the BenchMark was superior on NSGA-II

2- Old-GMHS :

This algorithm is a modified version of GMHS-BenchMark but it has been named with this name for some reason !

---- because of the bad results of GMHS-BenchMark before the last modification that has been illustrated above (inserting Xnew in the new HM any way),

We wanted to develop our own MO harmony searching algorithm.

So we introduced the following modifications to the old pseudocode:

1- Applying “selection operator using 2-tournament” on HM before improvisation

---

HM = selectionOperator(HM)

We notice that the HM will loose many solutions by SelectionOperator, and it will repeat some solutions

2- For Improvisation:

We returned to the original HS improvisation algorithm then developed it:

Just The cross probability Pc and gaussian mutation are taken from GMHS

New added:

- Objectives decompositions
- Extraction new HM method
- Damping band width

- New improvisation pseudocode:

```
1-  Create pdf for each objective:
2-  for h=1:HMS
3-     Xnew=[]
4-     for j=1:solDim
5-        if rand < HMCR
6-           if rand < Pc // Pc from GMHS-BenchMark
7-              randIdx=round(unifrnd(1,HMS))
8-              Xnew(j)= HM(randIdx,j)
9-           else
10-             Xnew(j)= HM(h,j)
11-          end
12-       else
13-          Xnew(j)=unifrnd(LB(j),HB(j))
14-       end
15-       if rand< PAR
16-          if rand< Pbw
17-                      obj=randi(gmhs.nObj)
18-                      r1= roulette_wheel(pdf(:,obj))
19-                      Xnew(j) = Xnew(j) +  rand*(HM(r1)-Xnew(j))
20-          else
21-           if rand<0.5
22-             Xnew(j) = Xnew(j)+rand*bw;
23-           else
24-             Xnew(j) = Xnew(j)-rand*bw;
25-          end
26-               end
27-       end
28-    end % end for j
29- Xnew=GaussianMutation(Xnew,Pgm, sigma)
30- Xnew=CheckBounds(Xnew, LB, HB);
31- newHM(h,:)=Xnew;
32- updat pdf
33- bw= bw * dampingBW
34- sigma = sigmadampingBW
35-  end // end for h
```

3- The modification on “Extracting new Harmony”

For sols in the last rank

Select based on crowding distance with probabilty 0.5

Select based on solution space distance with probabilty 0.5

End

the results of this modification have almost been equivelant to NSGA-II, but they were not better.

3- Developed GMHS:

This version is the Developed "old-GMHS" version

The new modifications :

1- cancelling the Selection operator before the improvisation

2- for improvisation

New added:

- Leader from EA

The improvisation pseudo code:

```
Inputs:
HMCR, PAR, PC, globalSearchSwitch,bw,Pgm,sigma, dampingBW
Outputs:
newHM
Start:
        Pdf =  normalize(1/HM.solutionsObjectiveValues)

        for h=1:size(HM)
            Xnew=HM(h)
            for j=1:length(HM(h))
                 if rand<HMCR
                        r=randi(size(HM))
                         if rand<PC
                            Xnew(j)=HM(r,j)
                         else
                            Xnew(j)=Xnew(j)
                  else
                        Xnew(j)=unifrnd(search space)
                   End
                If rand<PAR

                        if rand < globalSearchSwitch
                                if rand<exp(1-10*(iter/ numberOfIterations)
                                        obj = randomObjective
                                G= rouletteWheel(pdf)
                                        diff=gmhs.HM(G).sols(j)-Xnew(j)
                                        sigma=abs(diff/10)
```

```
                                Xnew(j)=normrnd(gmhs.HM(G).sols(j),sigma)
                        Else
                                Leader= select random harmony from EA
                                diff=Leader(j)-Xnew(j);
                                sigma=abs(diff /10);
                                Xnew(j)=normrnd(Leader(j),sigma);
                        end
                else
                        If rand<0.5
                                Xnew(j)=Xnew(j)+rand*bw
                         else
                                 Xnew(j)=Xnew(j)-rand*bw
                        end
                  end
            end
        end
         Xnew=gaussianMutation(Xnew,Pgm,sigma)
        bw=bw*dampingBW
        sigma=sigma*dampingBW
         newHM(h)=Xnew
         newObj=objFun(Xnew)
         update pdf
      end
end
```

3- Extraction new HM pseudocode:

```
4- Inputs:
5- newHM … (HMS*2), HMS
6- Outputs:
7- HM … (HMS)
8- Start:
9- For sols in the last rank
10-             idx =  find max(crowding Distance)
11-             if length(idx) >1 // if there are more than one solution have the best
    crowding distance
12-                 idx= find max(gridFitness(idx)) // select based on grid occupancy
```

```
13-                    if length(idx)>1 // if there are more than one solution have the best grid fitness
14-                        idx= find max(solSpaceDist(idx)) // select based on solutions space distance
15-                    End
16-                end
17-            end
18-            End
```

Algorithms parameters for all experiments

1- Population size = 50
2- Iterations = 25

The special Parameters of harmony searching algorithms for all experiments

1- HMCR = 0.99
2- Pc = 0.1
3- Pbw = 0.9 // global search probability
4- PAR = 0.9 // for developed and old, because it is adaptive in the benchmark
5- BandWidth = (HB-LB)*0.05
6- dampingBW = 0.99 // for developed and old only
7- sigma_GaussianMutation = (HB-LB)/20
8- Pgm=1/solDim if solDim~=1, else Pgm = 0.1 // gaussian mutation probability

The results:

To generate figures→ go to visualization folder

Results summary

objFun	Set Coverage (from best to worst)	GD (from best to worst)	HV (from best to worst)	NDS (from best to worst)	Delta (from best to worst)
SCH1	devloped	devloped	NSGA-II	same	NSGA-II
	NSGA-II	NSGA-II	devloped		devloped
	benchMark	benchMark	Old-GMHS		Old-GMHS
	Old-GMHS	Old-GMHS	benchMark		benchMark
SCH2	devloped		devloped	same	benchMark
	NSGA-II, benchMark		benchMark		NSGA-II
	Old-GMHS		NSGA-II		devloped
			Old-GMHS		Old-GMHS
FON	benchMark	benchMark	NSGA-II	same	NSGA-II
	devloped	devloped	benchMark		devloped
	NSGA-II	Old-GMHS	devloped		benchMark
	Old-GMHS	NSGA-II	Old-GMHS		Old-GMHS
POL	benchMark	benchMark	benchMark	same	benchMark
	devloped	devloped	NSGA-II		devloped
	NSGA-II		devloped		NSGA-II
	Old-GMHS	Old-GMHS	Old-GMHS		Old-GMHS
KUR	devloped	devloped	devloped	Devloped, benchMark, Old-GMHS	devloped
	benchMark	benchMark	benchMark		benchMark
	Old-GMHS	NSGA-II	NSGA-II		Old-GMHS
	NSGA-II	Old-GMHS	Old-GMHS	NSGA-II	NSGA-II
ZDT1	benchMark, NSGA-II	Devloped, benchMark, NSGA-II	NSGA-II	same	NSGA-II
	devloped		benchMark		devloped
	Old-GMHS	Old-GMHS	devloped		benchMark
			Old-GMHS		Old-GMHS
ZDT2	NSGA-II	Devloped, benchMark, NSGA-II	NSGA-II	same	NSGA-II
	devloped		benchMark		benchMark
	benchMark		devloped		devloped
	Old-GMHS	Old-GMHS	Old-GMHS		Old-GMHS
ZDT3	devloped	NSGA-II	NSGA-II	same	devloped
	benchMark	devloped	devloped		NSGA-II
	NSGA-II	benchMark	benchMark		benchMark
	Old-GMHS	Old-GMHS	Old-GMHS		Old-GMHS

ZDT4	devloped	devloped	devloped	Devloped, benchMark, Old-GMHS	devloped
	benchMark	benchMark	benchMark		benchMark
	Old-GMHS	NSGA-II	Old-GMHS		NSGA-II
	NSGA-II	Old-GMHS	NSGA-II	NSGA-II	Old-GMHS
ZDT6	devloped	devloped	NSGA-II	same	devloped
	benchMark	benchMark	benchMark		benchMark
	NSGA-II	NSGA-II	devloped		Old-GMHS
	Old-GMHS	Old-GMHS	Old-GMHS		NSGA-II

Conclusion :

- The developed-GMHS superiority ratio in terms of “setCoverage” is 60 %
- The developed-GMHS superiority ratio in terms of “GD” is 60 %
- The developed-GMHS superiority ratio in terms of “HV” is 30 %
- The developed-GMHS superiority ratio in terms of “NDS” is 100 %
- The developed-GMHS superiority ratio in terms of “delta” is 40 %

- The GMHS-benchMark superiority ratio in terms of “setCoverage” is 30 %
- The GMHS-benchMark superiority ratio in terms of “GD” is 20 %
- The GMHS-benchMark superiority ratio in terms of “HV” is 10 %
- The GMHS-benchMark superiority ratio in terms of “NDS” is 100 %
- The GMHS-benchMark superiority ratio in terms of “delta” is 20 %

- The old- GMHS ratio in terms of “setCoverage” is 0 %
- The old- GMHS superiority ratio in terms of “GD” is 0 %
- The old- GMHS superiority ratio in terms of “HV” is 0 %
- The old- GMHS superiority ratio in terms of “NDS” is 100 %
- The old- GMHS superiority ratio in terms of “delta” is 0 %

- The NSGA-II superiority ratio in terms of “setCoverage” is 20 %
- The NSGA-II superiority ratio in terms of “GD” is 10 %
- The NSGA-II superiority ratio in terms of “HV” is 60 %
- The NSGA-II superiority ratio in terms of “NDS” is 80 %
- The NSGA-II superiority ratio in terms of “delta” is 40 %

The end

Eng. Mohammad Sakka