Walchand College of Engineering, Sangli
Department of Information Technology

A
Presentation on

Design and Implementation of Rao's Algorithm

Presented By
1.Shraddha Varma (2017bteit00203)
2.Shreya Bodhane (2017bteit00204)
3.Sumit Lohar (2017bteit00207)

Guided by:
Dr. A. J. Umbarkar

Group No :15

Problem Statement

- Design & Implementation of Rao’s Algorithm for Solving Optimization Problem.

Objectives

- To Design spark based model of Rao's Algorithm. (100% completed in 1 sem)
- To Implement Rao's Algorithm in sequential and Distributed Manner. (100% completed in 1 sem & Distributed is 3 node hadoop spark cluster)
- Apply Rao's Algorithm to solve Optimization Problem.(100% Completed but remain Distributed Manner in 2 sem)

Introduction

- RAO Algorithm

  Type of Evolutionary Algorithm which is population based Iterative algorithm. Three equation given namely RAO-1,RAO-2 & RAO-3

- Multi-Demand Multi Dimensional Knapsack Problem

  This Problem we used for solve using Rao Algorithm. It is extension the classical Knapsack problem in which knapsack has a set of dimension

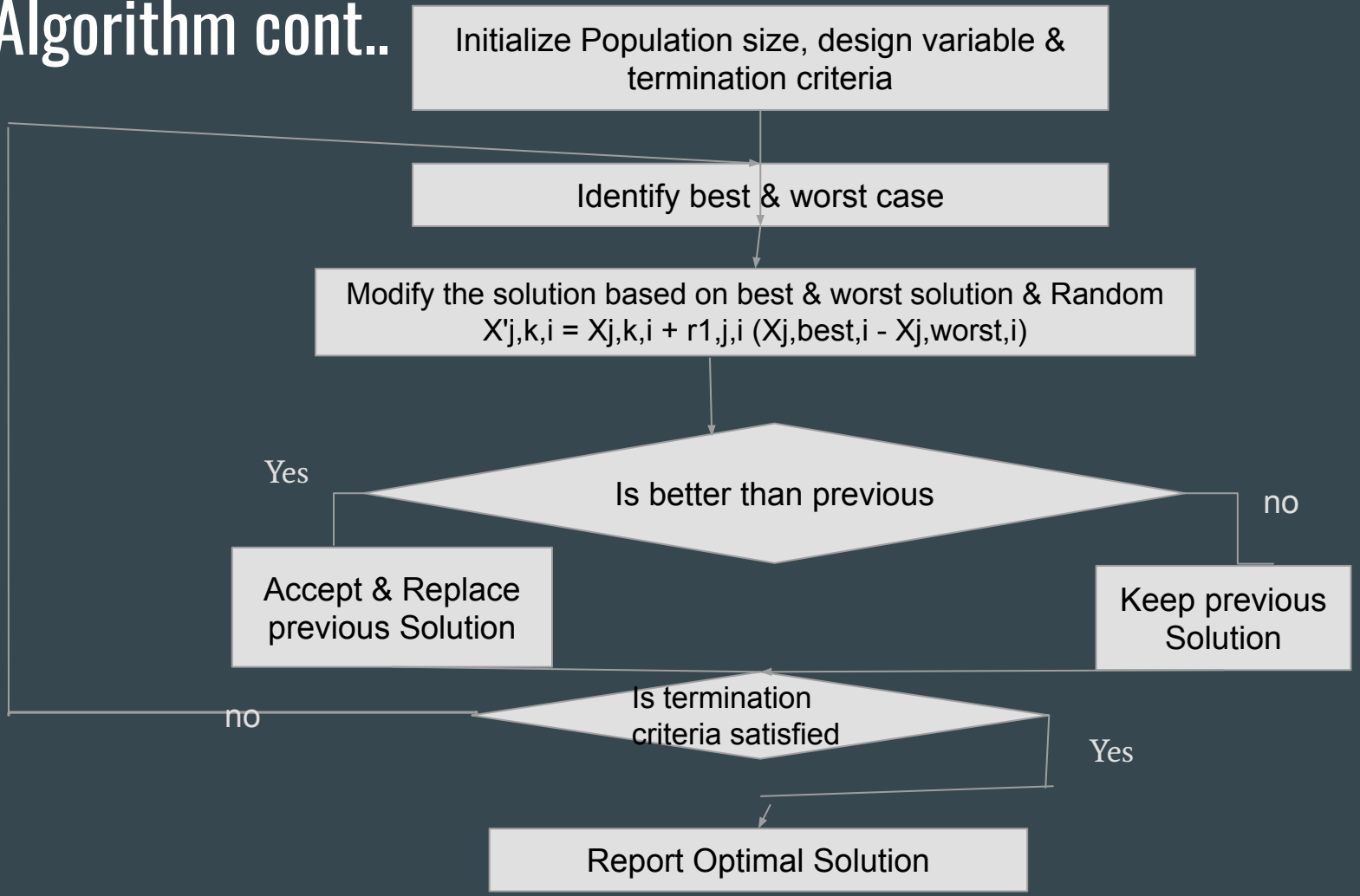
- SAMP Algorithm

  Self-adaptive Multi-population upgrade by using Rao's with the multi-population search process to enhance the diversity of search.

RAO Algorithm

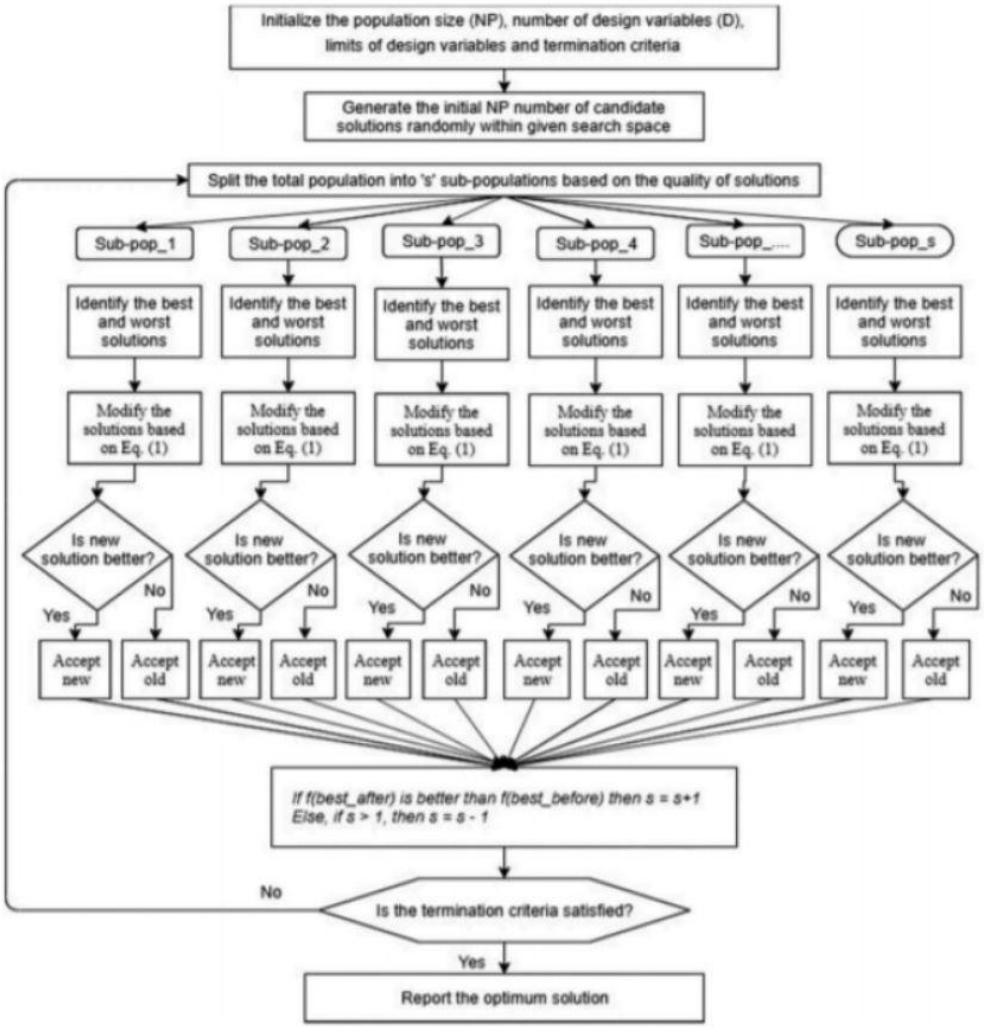
- It is simple population based Iterative algorithm where given design variable & number of candidate & we have to obtain best case & worst case value using objective function at each iteration upto given no of iterations
- RAO-1 : $X'_{j,k,i} = X_{j,k,i} + r1_{,j,i} (X_{j,best,i} - X_{j,worst,i})$
- RAO-2 : $X'_{j,k,i} = X_{j,k,i} + r1_{,j,i} (X_{j,best,i} - X_{j,worst,i}) + r2_{,j,i} (| X_{j,k,i} \text{ or } X_{j,l,i} | - | X_{j,l,i} \text{ or } X_{j,k,i} |)$
- RAO-3 : $X'_{j,k,i} = X_{j,k,i} + r1_{,j,i} (X_{j,best,i} - | X_{j,worst,i} |) + r2_{,j,i} (| X_{j,k,i} \text{ or } X_{j,l,i} | - (X_{j,l,i} \text{ or } X_{j,k,i})$
- These three algorithms are based on the best and worst solutions in the population and the random interactions between the candidate solutions.

RAO Algorithm cont..



Self-adaptive Multi-population Rao Algorithms

- Rao algorithms are upgraded with the multi-population search process to enhance the diversity of search.
- The number of sub-populations is changed adaptively considering the strength of solutions to control the exploration and exploitation of the search process.
- SAMP RAO-1 : $X'_{j,k,i} = X_{j,k,i} + r1_{j,i} (X_{j,best,i} - X_{j,worst,i})$
- SAMP RAO-2 : $X'_{j,k,i} = X_{j,k,i} + r1_{j,i} (X_{j,best,i} - X_{j,worst,i}) + r2_{j,i} (| X_{j,k,i} \text{ or } X_{j,l,i} | - | X_{j,l,i} \text{ or } X_{j,k,i} |)$
- SAMP RAO-3 : $X'_{j,k,i} = X_{j,k,i} + r1_{j,i} (X_{j,best,i} - | X_{j,worst,i} |) + r2_{j,i} (| X_{j,k,i} \text{ or } X_{j,l,i} | - (X_{j,l,i} \text{ or } X_{j,k,i})$


Initialize the population size (NP), number of design variables (D), limits of design variables and termination criteria
Generate the initial NP number of candidate solutions randomly within given search space
Split the total population into 's' sub-populations based on the quality of solutions
Sub-pop_1
Sub-pop_2
Sub-pop_3
Sub-pop_4
Sub-pop_....
Sub-pop_s
Identify the best and worst solutions
Modify the solutions based on Eq. (1)
Is new solution better?
Yes
No
Accept new
Accept old
If f(best_after) is better than f(best_before) then s = s+1
Else, if s > 1, then s = s - 1
Is the termination criteria satisfied?
No
Yes
Report the optimum solution

Multi Demand Multi Dimensional KnapSack Problem

- Classical 0-1 Knapsack problem.

  Maximize $\Sigma^{n}\square_{=1}$ pjxj

  subject to $\Sigma^{n}j_{=1}$ wixi $\leq$ C where Xj Ɛ {0,1} (j=1,.....,N)

- Adding it into multiple Capacities vectors C with M dimension, q is demand, less-than-or-equal-to inequalities & greater-than-or-equal-to inequalities
- Problem defined as :

$$\text{Maximize } \Sigma^{n}_{i=1} p_i x_i$$
$$\text{subject to } \Sigma^{n}_{i=1} w_i x_i \leq C^k \quad (k=1,.....,m)$$
$$\Sigma^{n}_{i=1} w_i x_i \geq C^k \quad (k=1+m,.....,m+q)$$
$$\text{where } Xi \text{ Ɛ } \{0,1\} \quad (i=1,.....,N)$$

Parameters

- N the number of items
- p_j the profit of item j
- w_j the weight of item j
- c the capacity of a single knapsack
- m the number of knapsack constraints
- w^k_j the weight of item j in knapsack k
- c^k the capacity of knapsack k
- q the number of demand constraints
- n the number of groups

Maximize $z = 10x1 + 20x2 + 30x3 + 40x4 + 50x5 + 60x6 + 70x7 + 80x8$

Subject to

Dimensional constraints

(1) $5x1 + 20x2 + 25x3 + 35x4 + 40x5 + 45x6 + 55x7 + 60x8 \leq 150$

(2) $90x1 + 120x2 + 70x3 + 110x4 + 90x5 + 65x6 + 80x7 + 150x8 \leq 300$

Demand constraints

(3) $5x1 + 20x2 + 100x3 + 35x4 + 60x5 + 45x6 + 50x7 + 60x8 \geq 80$

(4) $90x1 + 60x2 + 70x3 + 110x4 + 90x5 + 45x6 + 20x7\ 10x8 \geq 200$

$$xj \ \varepsilon \ \{ 0,1\} \text{ for } j = 1,..., 8.$$

OPTIMAL SOLUTION BIT STRING IS (0 0 0 1 1 0 1 0) with $z = 160$.

Results

```
shreya@shreya-Lenovo-ideapad-320-15IKB: ~/Downloads
File  Edit  View  Search  Terminal  Help
Best=0.000384828717489
xbest=[3.5838606571145606, -1.842973747555701]
The objective function value = 0.000384828717489
The optimum values of variables = [3.5838606571145606, -1.842973747555701]
shreya@shreya-Lenovo-ideapad-320-15IKB:~/Downloads$ python rao.py
Best=0.000188020895426
xbest=[3.584824323377416, -1.8447144321280626]
The objective function value = 0.000188020895426
The optimum values of variables = [3.584824323377416, -1.8447144321280626]
shreya@shreya-Lenovo-ideapad-320-15IKB:~/Downloads$
```

Fig 1.Single Pc with Sphere Function

```
    0.0644    -0.0185     4.4958

Iteration No. = 2
%%%%%%%% Final population %%%%%%%%
   1.0e+03 *

   -0.0257    -0.0141     0.8617
   -0.0110     0.0258     0.7871
    0.0593     0.0081     3.57924

   -0.0410     0.0362     2.9858
    0.0644    -0.0185     4.4958

Iteration No. = 3
%%%%%%%% Final population %%%%%%%%
   1.0e+03 *

   -0.0257    -0.0141     0.8617
   -0.0110     0.0258     0.7871
    0.0216     0.0409     2.1413
   -0.0410     0.0362     2.9858
    0.0055    -0.0031     0.0393

Iteration No. = 4
%%%%%%%% Final population %%%%%%%%
   1.0e+03 *

   -0.0257    -0.0141     0.8617
   -0.0069     0.0086     0.1206
    0.0326     0.0172     1.3595
    0.0014     0.0073     0.0554
    0.0055    -0.0031     0.0393

Iteration No. = 5
```

Fig 2.Result with Iteration 3 using Sphere Function

```
Iteration No. = 3000
%%%%%%%% Final population %%%%%%%%%
  1.0e-11 *

  Columns 1 through 18

   -0.0131    0.0116   -0.0057   -0.0344    0.0917   -0.1104    0.0157   -0.1381    0.1484    0.0570    0.1467    0.0492    0.1030   -0.0995   -0.0963
    0.0643    0.0429    0.0080   -0.0347    0.0621   -0.0472    0.0086   -0.1187    0.1637    0.0993    0.2243    0.0137    0.1299   -0.0766   -0.1797
   -0.0046    0.1291    0.0144   -0.0097    0.0087   -0.1761    0.0298   -0.0532    0.1633    0.0383    0.1687    0.0121    0.0647   -0.0524   -0.0830
    0.0511    0.1479    0.0075   -0.0609    0.0763   -0.2597   -0.0214   -0.0430    0.1794    0.1017    0.1648    0.0030    0.1282   -0.0968   -0.1550
   -0.0308    0.0934    0.0083   -0.0358    0.1244   -0.0124   -0.0155   -0.2037    0.2265    0.0591    0.2040   -0.0550    0.0849   -0.0466   -0.1016
    0.1304   -0.0373   -0.0151   -0.0615    0.1104   -0.1182    0.0791   -0.0891    0.1549    0.0934    0.1940   -0.0048    0.1091    0.0536   -0.2912
    0.0087    0.0426   -0.0037   -0.0180    0.0825   -0.0911    0.0353   -0.1240    0.1815    0.0820    0.1747    0.0428    0.0999   -0.0784   -0.1416
    0.0944    0.1192    0.0043   -0.0330    0.0797   -0.0876    0.0286   -0.0687    0.2019    0.0735    0.2167   -0.0242   -0.0212    0.0498   -0.1805
    0.0746    0.1033   -0.0035   -0.0055    0.0320   -0.1026    0.0243   -0.1408    0.2088    0.0618    0.2058   -0.0208   -0.0309   -0.0986   -0.1591
    0.0514    0.0699   -0.0108    0.0085    0.0298   -0.3115    0.0611   -0.0482    0.1780    0.0699    0.1589   -0.0276    0.0614   -0.0360   -0.0876

  Columns 19 through 31

   -0.1313   -0.2336    0.2300   -0.5181    0.0640    0.1231    0.0830    0.2282    0.0421   -0.1190    0.0067   -0.3721    0.0000
   -0.0548   -0.1497    0.2198   -0.4902    0.0365    0.1043    0.1362    0.1454   -0.0415   -0.1163   -0.0162   -0.3718    0.0000
   -0.0970   -0.2481    0.1417   -0.5345    0.1207    0.0479    0.1278    0.1796   -0.0488   -0.1718    0.0969   -0.3658    0.0000
   -0.0265   -0.2343    0.1878   -0.4941    0.0705    0.1300    0.1459    0.1809   -0.0545   -0.1149    0.0160   -0.3593    0.0000
   -0.0854   -0.0609    0.2080   -0.5071   -0.0035    0.0336    0.0821    0.0516    0.0732   -0.2070    0.0104   -0.4355    0.0000
   -0.0744   -0.1630    0.2214   -0.4931    0.0227    0.2407    0.1090    0.1548    0.0317   -0.1246   -0.0963   -0.3189    0.0000
   -0.1172   -0.1764    0.2120   -0.5065    0.0481    0.1334    0.1326    0.1392   -0.0221   -0.1191   -0.0399   -0.3427    0.0000
   -0.0301   -0.1500    0.2188   -0.4969    0.0547    0.1163    0.1334    0.1045   -0.0725   -0.1119    0.0566   -0.3482    0.0000
   -0.0289   -0.1824    0.3297   -0.5213    0.1019    0.0996    0.1060    0.0466   -0.0172   -0.0947    0.0409   -0.3414    0.0000
   -0.1901   -0.1561    0.0602   -0.5178    0.2167    0.1294    0.1670    0.1198   -0.0407   -0.0966    0.1044   -0.3453    0.0000

Optimum value = 1.022194521e-22
```

Fig 3.Distributed Result with Sphere Function using 3-Node Clusters

```
The objective function value = 0.0006235463964256088
The optimum values of variables = [3.5878826255085396, -1.8482237730479525]
```

ackley	1.1722266131898695
bealey	0.378043569406309
booth	1.0189905127176186
easom	-0.0380310946429861
matyas	0.028816147429688306
rastring	2.594765487968942
rosenbrock	2.3651639050585324e+18

Fig 4. BenchMark Function Results

-0.0111	-0.0072	0.0184	-0.0094	-0.0111	0.0043	0.0013	0.0117	-0.0254	0.0056	-0.0012	-0.000
-0.0138	-0.0055	0.0187	-0.0032	-0.0041	0.0050	0.0026	0.0165	-0.0265	0.0043	0.0052	-0.009
-0.0111	-0.0051	0.0192	-0.0064	-0.0147	0.0045	0.0053	0.0109	-0.0263	0.0031	0.0033	-0.005
-0.0108	-0.0072	0.0194	-0.0094	-0.0094	0.0042	0.0008	0.0122	-0.0250	0.0037	0.0025	-0.001
-0.0110	-0.0066	0.0189	-0.0025	-0.0132	0.0052	0.0008	0.0072	-0.0265	0.0001	0.0013	-0.007
-0.0104	-0.0073	0.0201	-0.0105	-0.0071	0.0037	0.0023	0.0112	-0.0254	-0.0002	0.0118	-0.001
-0.0111	-0.0056	0.0194	0.0015	-0.0135	0.0038	0.0022	0.0113	-0.0255	0.0029	0.0045	-0.004
-0.0111	-0.0043	0.0174	-0.0126	-0.0116	0.0042	0.0024	0.0071	-0.0248	0.0053	0.0097	-0.011
-0.0110	-0.0034	0.0176	-0.0079	-0.0116	0.0041	0.0025	0.0043	-0.0248	0.0053	0.0095	-0.012
-0.0109	-0.0001	0.0179	-0.0050	-0.0112	0.0020	-0.0035	0.0002	-0.0272	0.0082	0.0006	0.005

Columns 19 through 21

0.0053	-0.0241	3.9369
0.0040	-0.0237	3.9906
0.0046	-0.0246	4.0961
0.0059	-0.0241	3.8974
0.0099	-0.0256	4.1819
0.0079	-0.0248	4.1822
0.0125	-0.0247	4.1221
0.0079	-0.0256	4.2060
0.0082	-0.0256	4.0805
0.0128	-0.0254	4.2351

```
best=1759.798903
mean=2555.385902
worst=3897.392241
std. dev.=797.504266
mean Fes=496.000000
```

SAMP-1 Rao Algorithm

```
Columns 1 through 18

    0.0000   -0.0044   -0.0003   -0.0003    0.0000   -0.0063    0.0022    0.0004   -0.0006   -0.0031    0.0029
    0.0009   -0.0020    0.0005   -0.0004   -0.0003   -0.0002    0.0020   -0.0018   -0.0008   -0.0009    0.0061
    0.0001   -0.0018   -0.0001   -0.0009   -0.0010   -0.0008    0.0020    0.0000   -0.0005   -0.0018    0.0037
   -0.0004   -0.0015   -0.0008   -0.0005   -0.0017   -0.0019    0.0033    0.0002   -0.0005    0.0004    0.0039
    0.0028    0.0012   -0.0026   -0.0006   -0.0053   -0.0024    0.0022    0.0003   -0.0001   -0.0018    0.0043
    0.0009   -0.0014   -0.0011   -0.0003   -0.0003   -0.0011    0.0034    0.0009    0.0004    0.0010    0.0016
   -0.0015   -0.0021    0.0006    0.0015   -0.0028   -0.0016    0.0049    0.0005   -0.0001   -0.0023    0.0030
   -0.0011   -0.0025   -0.0000    0.0008    0.0017   -0.0006    0.0043   -0.0003    0.0001   -0.0012    0.0020
   -0.0004   -0.0018   -0.0003    0.0011   -0.0029   -0.0022    0.0036    0.0008   -0.0005   -0.0056    0.0042
   -0.0031   -0.0018    0.0019   -0.0020   -0.0057    0.0022    0.0030    0.0007   -0.0012   -0.0020   -0.0000

Columns 19 through 21

   -0.0026    0.0025    1.2339
   -0.0035   -0.0023    0.9896
   -0.0036    0.0019    0.9147
    0.0024   -0.0016    1.2118
   -0.0019    0.0010    1.1739
   -0.0034    0.0000    0.6870
   -0.0048   -0.0008    1.0700
   -0.0038    0.0004    1.0773
   -0.0039    0.0001    1.1525
   -0.0039   -0.0024    1.0594


best=4242.122822
mean=8022.446173
worst=13078.687399
std. dev.=2742.198304
mean Fes=439.000000
>>
```

SAMP-2 Rao Algorithm

```
Columns 1 through 18

    0.0094    0.0110    0.0008    0.0039   -0.0002    0.0016   -0.0052    0.0103    0.0109   -0.0036   -0.0049
    0.0133    0.0097    0.0046    0.0069   -0.0002    0.0032   -0.0042    0.0054    0.0115    0.0007   -0.0035
    0.0100    0.0126    0.0025    0.0056   -0.0002    0.0026   -0.0097    0.0108    0.0104   -0.0037    0.0035
    0.0076    0.0139    0.0012    0.0043   -0.0002    0.0033   -0.0013    0.0090    0.0100   -0.0110    0.0035
    0.0100    0.0118    0.0052    0.0025    0.0000    0.0068    0.0085    0.0122    0.0105   -0.0024   -0.0065
    0.0072    0.0126    0.0037    0.0075    0.0008    0.0077   -0.0018    0.0102    0.0107   -0.0061    0.0018
    0.0093    0.0111    0.0025    0.0066    0.0001    0.0111   -0.0062    0.0155    0.0103    0.0041   -0.0001
   -0.0020    0.0134    0.0043    0.0120    0.0014    0.0107   -0.0056    0.0050    0.0102   -0.0034    0.0093
    0.0007    0.0133    0.0031    0.0073    0.0005    0.0113   -0.0068    0.0113    0.0100   -0.0077    0.0075
    0.0023    0.0131    0.0041    0.0115    0.0008    0.0140   -0.0046    0.0086    0.0101   -0.0085   -0.0011

Columns 19 through 21

   -0.0053    0.0168    1.3770
    0.0002    0.0199    1.5179
   -0.0013    0.0161    1.5840
   -0.0019    0.0164    1.6676
   -0.0019    0.0162    1.6273
   -0.0012    0.0144    1.1695
   -0.0017    0.0169    1.6975
   -0.0069    0.0137    1.4961
    0.0019    0.0090    1.2749
    0.0027    0.0165    1.4430


best=918.656663
mean=1960.306625
worst=3933.624222
std. dev.=1181.054523
mean Fes=473.000000
>>
```

SAMP- 3 Rao Algorithm Distributed

References

1. R. V Rao,“Rao algorithm:three metaphor simple algorithm for solving optimization problem” of IJIEC vol 11,2020.
2. R. V Rao, K. C. More, and J. Taler, "Dimensional optimization of a micro-channel heat sink using Jaya algorithm." *Applied Thermal Engineering,* vol. 103 pp. 572-582, 2016.
3. K. V. Price, N. H. Awad, M. Z. Ali, P. N. Suganthan, "Problem Definitions and Evaluation Criteria for the 100-Digit Challenge Special Session and Competition on Single Objective Numerical Optimization," Technical Report, Nanyang Technological University, Singapore, November 2018.
4. Amine Lamine, Mahdi Khemakhem and Habib Chabchoub “Knapsack Problems involving dimensions, demands and multiple choice constraints: generalization and transformations between formulations”.

THANK YOU!!!