Improvement-of-Multi-Population ML-SHADE

ALGORITHM1 IMPROVEMENT-OF-MULTI-POPULATION ML-SHADE

<u>Notations</u>

C: the number of sub-populations

M_C^F,M_C^{CR}: the history memory of mean F, CR values of the c^{th} sub-population

A_C: the archive of inferior solutions of the c^{th} sub-population

pop: initial population

subpop: sub-populations

N^{init}: the size of the initial population

N^{sub} : the size of a sub-population

N^{min}:the lower bound of N^{sub}

MS_c: the mutation strategy of subpop

```
nfe = 0
pop = Initialize(N^init)
NP = |pop|
for c=1 to C do
   Randomly select a reference
point R from pop
     subpop[c] = {R}
     pop = pop \ {R}
end for
for c = 1 to C do
   for j = 2 to NP/C do
      Find the nearest individual x ∈
      pop to R
      subpop[c] = subpop[c] ∪ {x}
      pop = pop \ {x}
   end for
  Initialize values in M_C^F,M_C^CR
  Set A^c = Ø, MS^C = c%3
end for
while the termination criterion is
not satisfied do:
  for c = 1 to C do
     subpop[c] = Update(subpop[i],Ac,c,Mc,nfe)
   end if
  end for
```

```
if nfe > Max_nfe /2
    for c =1 to C do
      j = rand of top p% of subpop[c]
      for i =2 to N^sub
        if    the distance- between i -
and j are small enough
        subpop[c] = subpop[c]\{i}
      end if
      end for
      if subpop[c] size less than N^min
        subpop[c] size = N^min
      end if
    end for
end if
for c=1 to C do
    if subpop[c] best fitness not improved for 1000 generations
      subpop[c] reinitialization
    end if
end for
Update N^sub using the LPSRstrategy
for c = 1 to C do
  if N^sub < |subpop[c]|
  Resize subpop[c] by removing the
    worst individuals
  Resize Ac by removing individuals
    randomly
  end if
end for
end while
return the best solution in subpop
```

ALGORITHM2 UPDATE FUNCTION

Fuction Update(pop,A,k,M,nfe)

Output: : The result of evolving pop after one generation

Notations

pop: a population of individuals

A: the archive of inferior solutions

k: the index of the to-be-updated history memory

M: the history memory of mean F, CR values

nfe: the number of fitness evaluations already consumed

MAX_NFE: the maximum number of function evaluations

NP: the number of individuals in pop

H: the size of history memory

randn, randc, randu: normal, Cauchy, and uniform distribution

D: problem dimension

S : the archive of successful F, CR, and fitness improvement

scale: minimum and maximum values for scale

```
S_F = Ø ,S_CR= Ø , S_Δf= Ø
Scale = scale(min) +(scalemax -scalemin)×(nfe/MAX_NFE)
Newpop ={}
for i = 1 to NP do
   r = a random integral value in [1, H]
   F_i= randc(M_F,r, scale), CR_i= randn(M_CR,r , 0.1)
   Repair F_i and CR_i
   xi = pop[i]
   Generate a mutant vector vi
   Generate a trial vector ui
   if f(ui) ≤ f(xi) then
      newpop = newpop ∪ {ui}
      S_F = S_F ∪ {Fi}, S_CR = S_CR ∪ {CRi}, S_Δf = S_Δf ∪ {Δf_k},
      A = A ∪ {xi}
   else
      newpop = newpop ∪ {xi}
   end if
   nfe = nfe + 1
end for
if S_CR ≠ Ø and S_F ≠ Ø then
  Update the kth memory in M_F and M_CR,
  k = k mod H +1
  nn(nfe of no_improvement) = 0
else
  nn += NP
  if randu(0.0, 1.0) ≤ nn/ nfe) then
     M_CR,,k = randu(0.0, 1.0)
     M_F,k = randu(0.0, 1.0)
```

```
    nn =0
    k = k mod H +1
  end if
end if
return newpop
```

IMPROVEMENT-OF-MULTI-POPULATION ML-SHADE(IMPML-SHADE) ALGORITHM

A. Overview

In this paper we propose a improvement MPML-SHADE (IMPMLSHADE) algorithm. The pseudo code of IMPMLSHADE is on above. Three modifications are briefly described here:

I. Multiple mutation strategies: We appoint different mutation strategies to each sub-population. We have three different mutation strategies, which has different characters, and that may compensate each shortcoming.

II. Reinitialization: We fund it is common way to improve the algorithm by reinitialization. We have many sub-populations, so our reinitialization criteria for sub-population is when its best fitness does not improve for a while, and we keep it best individual, and restart other individuals.

III. Eliminate some individual in sub-population: We find that polynomial mutation in MPML-SHADE may reduce some performance of this algorithm, so we replace it with this way: we randomly choose one individual in sub-population, and if other individuals are too close to it, and we will eliminate them from this population.

B. Initialization

The initial population are generated by uniform random initialization within the range of variables, as the rule of CEC2022 competition requires.

C. Mutation Strategy

We use three different mutation strategies:

1. current-pbest/1 mutation strategy

   $vi = xi + F(x_{pbest} - xi) + F(x_{r1} - x_{r2})$

   xpbest is a solution picked randomly from the top p% of the population based on fitness value.

2. current-to-rand/1 mutation strategy

   $vi = xi + F(x_{r1} - x_{r2})$

3. weighted-rand-to-qbest/1 mutation strategy

   $vi = F \cdot xi + F \cdot F_a \cdot (x_{qbest} - x_{r2})$

   xqbest is a solution picked randomly from the top q% of the population based on fitness value.

$$q = 2p - p \cdot (\frac{nfe}{Max_{nfe}})$$

$$F_a = 0.5 + 0.5 \cdot (\frac{nfe}{Max_{nfe}})$$

When the $v_{j,i}$ of dimension j in the mutant vector vi goes outside of the feasible range[$x_{min}.x_{max}$], we repair the value by below.

$$Vj,I = \begin{cases} \frac{xmin+xj,i}{2}, & if\ v_{j,i} < x_{min} \\ \frac{xmax+xj,i}{2}, & if\ v_{j,i} > x_{max} \end{cases}$$

D. Crossover

The trial vector ui is generated by binomial crossover, j_{rand} is a random integral value in [1, D].

$$u_{j,i} = \begin{cases} v_{j,i}\ if\ rand[0,1) \leq CR\ or\ j = j_{rand} \\ \qquad\qquad x_{j,i} \end{cases}$$

E. Selection

The trial vector ui will replace the target vector xi when its fitness is better than xi's fitness.

F. Parameter Control

This algorithm is based on MPMP-SHADE, and it use its success history-based adaption and LPSR strategies. The history memory M_F and M_{CR} store potential mean values of F and CR. We use same way like MPML-SHADE to repair mechanism: when CR is under zero, we set it to the absolute value; when CR is up than one, we set it to one. We update the history memory in the same way as LSHADE does.

$$\Delta f_k = |f(ui) - f(xi)|$$

$$w_k = \frac{\Delta fk}{\sum_{l=1}^{|S|} \Delta fl}$$

$$mean_{WL}(S) = \frac{\sum_{k=1}^{|S|} w_k \cdot S_k^2}{\sum_{k=1}^{|S|} w_k \cdot S_k}$$

G. Terminating Criterion

Our algorithm stops when the maximum number of fitness function evaluations (Max_nfe) is reached.

EXPERIMENTS AND RESULTS

A. Benchmark Functions

There are 12 test functions f1 – f12 with two dimensions: D = 10 and 20 in the CEC2022 Single Objective Bound Constrained Optimization Competition. The search ranges of decision variables are limited in [-100, 100] for all functions.

B. Parameter Setting

The parameter that need to be set is list below :

1. N^{init} (size of the initial population) is set 3.6D·C
2. N_{min} (minimal sub-population size) is set 4
3. H (size of the success history memory) is set 6
4. r^{arc} (archive size |A|) is set 2.6
5. p (required in mutation) is set 0.11
6. M_C^F,M_C^{CR}(initial values of cth sub-population's F/CR memory) all set 0.5
7. C (the number of sub-populations) is set D
8. $scale_{min}$ (required in LSPI) is set 0.1
9. $scale_{max}$ (required in LSPI) is set 0.2
10. stuck (for reinitialize criteria) is set 1000

C. Algorithm Complexity

It's defined by CEC2022 competition.

The experiments were performed in a Windows 10 environment with Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz and 8GB DDR4 RAM. The IMPML-SHADE was set by C++.

	T0(ms)	**T1(ms)**	**T2(ms)**	**(T2-T1)/T0**
D=10	15	394.2	934.4	36.01
D=20	15	833.2	1921.6	72.56

D. Results

D = 10, the Max_nfe is 200000.D = 20, the Max_nfe is 1000000. Each test function runs for 30 times. According to the competition rules, statistics including the best, the worst, median, mean, and standard deviation of the error values over 30 runs are reported in below table.

D = 10

Func.	**Best**	**Worst**	**Median**	**Mean**	**Std**
1	0	0	0	0	0
2	0	0.0071236	0.000282	0.00123	0.001773
3	3.20E-06	0.000146	2.08E-05	2.51E-05	2.80E-05
4	2.009834	5.9854882	3.99484	4.024889	0.970414
5	0	0	0	0	0
6	0.07302	1.2764691	0.360895	0.45126	0.349045
7	7.08E-07	0.0134633	2.32E-05	0.000567	0.002411

8	0.097706	2.3269222	0.321994	0.59542	0.563649
9	229.2844	229.28438	229.2844	229.2844	0
10	0.125061	100.35764	12.11255	26.72185	34.23599
11	0	0	0	0	0
12	158.6184	160.69686	159.1337	159.131	0.51456

D = 20

Func.	Best	Worst	Median	Mean	Std
1	0	3.51E-07	0	3.76E-08	6.84E-08
2	0.199591	5.344211	2.860547	2.552853	1.488691
3	9.65E-06	0.00011	3.35E-05	4.14E-05	2.69E-05
4	5.803387	10.66196	7.355454	7.595453	1.263993
5	0	0	0	0	0
6	10.36967	45.34952	23.61067	24.18864	6.806501
7	1.604734	22.52121	15.24916	14.38089	6.29564
8	3.45966	21.45848	20.64166	18.31117	4.556371
9	180.7813	180.7813	180.7813	180.7813	1.86E-13
10	0.063286	30.77597	1.1769	8.124159	10.15385
11	0	16.12686	1.424377	2.413778	3.605409
12	230.6691	233.8578	232.1725	232.2804	0.797192