L-RAM-JAPDE and H-RAM-JAPDE: Deterministic Population Size Reduction Differential Evolution Algorithms Based on RAM-JAPDE

Andrés Ramos, Miguel León
School of Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden
ari19001@mdh.se, miguel.leonortiz@mdh.se

***Abstract*—RAM-JAPDE is an adaptive differential evolution algorithm which applies RAM, an adaptive method that ranks the individuals of the population and keeps a record of the performance of the mutation strategies applied to each group combined with JAPDE, an adaptive method that adapts F and CR parameters jointly. This paper proposes two algorithms that adapt PS by applying Population Size Reduction methods based on RAM-JAPDE. These algorithms are L-RAM-JAPDE, that uses Linear Population Size Reduction, and H-RAM-JAPDE, that uses Reduction to the Half Population Size Reduction. These algorithms were evaluated using the benchmark functions proposed at CEC2014 world congress. L-RAM-JAPDE and H-RAM-JAPDE were compared to other state-of-the-art using Wilconxon statistical tests and two N to N comparison metrics. The experimental results proves that the first approach improves its precursor and that both are competitive or even outperforms the counterparts, which indicates that other state-of-the-art algorithms could improve their performance by applying a population size reduction method.**

***Index Terms*—Differential evolution, global optimization, adaptive parameter, population size reduction, evolutionary optimization.**

I. Introduction

DIFFERENTIAL Evolution (DE) is a new Evolutionary Algorithm (EA) proposed by Storn [1]. DE is effectively used for many real-world applications [2] where the optimization of parameters is required. This is due to its easy implementation combined with good results.

Since DE is a genetic population based algorithm, its performance is heavily dependant on two phases that occur in each generation: mutation and crossover. In addition, the performance also depends on three main parameters: F that affects the magnitude of the changes of the mutated vector, CR that affects the magnitude of the crossover vector and the population size PS [3]. A wrong tuning of the previous parameters could provoke the two main issues of DE, which are slow convergence and stagnation into local optima. On the one hand, if the individuals are too diverse and does not exploit any region on the continuous space, it can cause that it requires inviable amount of time to find a good enough solution. On the other hand, if the algorithm uses a greedy approach to exploit a region and does not explore other regions of the search space can cause to never find a global optima, therefore, stagnate into a local optima.

In order to have a good balance regarding the trade-off between exploration and exploitation, different algorithms have been implemented by trying out new mutation and crossover operators and different ways of configuring the control parameters. One of them, presented as L-SHADE [4], an extended version of SHADE where the parameter PS is modified over run-time, won the competition presented in CEC2014 [5]. This fact, suggested that it might be worthwhile trying to apply mechanisms to control parameter PS overtime.

This paper proposes L-RAM-JAPDE and H-RAM-JAPDE, two algorithms that result of applying a method of reducing the population in a dynamical way over time based on RAM-JAPDE[6]. The first one, L-RAM-JAPDE uses a linear reduction, the same as L-SHADE. The second one, H-RAM-JAPDE uses reduction to the half method. Employing a Population Size reduction method could be beneficial to avoid the issues in DE, since it allows to have bigger values of PS at the beginning, which gives a better exploration in early stages of the search, and lower values at the end, which allows a better exploitation at later stages of the search. The aim of this work is to show that it is worth trying to use a method that modifies the control parameter PS on any algorithm.

The remaining document is organized as follows. First, Section II explains the fundamentals of the original differential evolution algorithm and then the bases of RAM-JAPDE. In Section III, a presentation of the related work regarding to adaptation of control parameters is done. Both approaches based on RAM-JAPDE presented in this paper are described in Section IV. Then, a comparison between the use of different parameters, the use of population size reduction method and state-of-the-art algorithms are made in Section V. Finally, Sections VI and VII comprehend the conclusion of the study and the possible future work in the line of this paper.

II. Background

In this section, the fundamentals of DE and the precursor algorithm is explained. This is important in order to understand the mechanisms of the two new methods presented in this paper. First of all, we will explain the base algorithm of the Differential Evolution[1] and then we will explain the strategy used as the precursor[6] algorithm used in L-RAM-JAPDE and

H-RAM-JAPDE. Also, the terminology in order to follow up with the future explanations is described.

A. Differential Evolution

Differential Evolution, proposed by R. Storn and K.Price in 1995[1] consists of a population represented by a set of parameters vectors $v_i = (x_1, ..., x_D), i = 1, ..., PS$ where PS is the population size and D is the dimension of the target problem. Each value of the vector is initialized with a real random value according to a uniform distribution with values included inside the range $[x^{MIN}, x^{MAX}]$, where x^{MIN} and x^{MAX} define the boundaries of the search space for the corresponding problem.

After the initialization phase, the population is evolutionary modified with three operations that comprehend the cycle of the DE. These three operations are Mutation, Crossover and Selection. After each cycle of the loop, a new generation is obtained, which is the population for the next DE cycle until we reach a termination criterion, e.g after managing to create the generation number X. These DE operations are the following ones:

- **Mutation:** In this first phase, the generation of a mutant population is created. This population is notated as V whose individuals' vectors of the generation G are notated as $v_{i,G}, i \in 1...PS$. This population is generated as a result of applying a mutation strategy to each individual of the population X. Some examples of these mutation strategies are the following expressions:
  - rand/1

$$v_{i,G} = x_{r1,G} + F * (x_{r2,G} - x_{r3,G}) \tag{1}$$

  - rand/2

$$v_{i,G} = x_{r1,G} + F * (x_{r2,G} - x_{r3,G} + x_{r4,G} - x_{r5,G}) \tag{2}$$

  - best/1

$$v_{i,G} = x_{best,G} + F * (x_{r1,G} - x_{r2,G}) \tag{3}$$

  - best/2

$$v_{i,G} = x_{best,G} + F * (x_{r1,G} - x_{r2,G} + x_{r3,G} - x_{r4,G}) \tag{4}$$

  - current-to-rand/1

$$v_{i,G} = x_{i,G} + F \cdot (x_{r1,G} - x_{i,G} + x_{r2,G} - x_{r3,G}) \tag{5}$$

  - current-to-best/1

$$v_{i,G} = x_{i,G} + F * (x_{best,G} - x_{i,G} + x_{r2,G} - x_{r3,G}) \tag{6}$$

  where $x_{r1,G}, ..., x_{r5,G}$ are uniformly random selected individuals from the population X in the current generation. $x_{best,G}$ is the best individual in the population X of the generation G. F stands for the scaling factor, which falls in the real interval [0,2] and it is used to control the impact of the change from the original vector to the mutant one.
- **Crossover:** In this phase, for each individual in the population a new vector is generated from the result of the combination of the parent's vector and his mutant vector. There are different ways of accomplishing crossover such as binomial crossover and exponential crossover. In this case, we will focus on explaining the binomial (used in our proposed algorithms), which uses the following expression:

$$u_{j,i,G} = \begin{cases} v_{j,i,G} & \text{if } rand[0,1] \leq CR \text{ or } j = j_{rand} \\ x_{j,i,G} & \text{otherwise} \end{cases} \tag{7}$$

  where $rand[0,1]$ represents a uniformly generated random real number from $[0,1]$. j_{rand} is a uniformly randomly selected integer value from $[1, D]$, which ensures that each trial vector $u_{i,G}$ differs at least in the j_{rand} dimension from its mutant vector $x_{i,G}$ in the population. CR stands for the crossover rate, a parameter with a value that falls within the interval [0,1] that combined with rand will determinate wether the mutant value is chosen or discarded.
- **Selection:** In this final phase, a selection of the best individuals is made. In this step, each individual from the population X is compared one to one against the individual of the population U. The best ones are being kept, which will generate the population used for the next generation, $X_G + 1$. In order to carry out this operation, the following expressions are being used:

$$x_{i,G+1} = \begin{cases} u_{i,G} & \text{if } f(u_{i,G}) \leq f(x_{i,G}) \\ x_{i,G} & \text{otherwise} \end{cases} \tag{8}$$

  where $f(x)$ is a function called fitness function. This function is used to evaluate the fitness of an individual, which is calculated with the error of the proposed individual regarding the optimal solution of the problem. Consequently, the individuals with lower fitness value are the better solutions to the target problem, since they present less error.

B. RAM-JAPDE

The methods presented in this paper (L-RAM-JAPDE and H-RAM-JAPDE) are based on the enhancing of RAM-JAPDE[6], which integrates the method of Rank-Based Adaptation of Mutation Strategies from RAM and the Joint Adaptation of Parameters in DE from JAPDE.

RAM [7], as many others methods, creates a system where the mutation strategies that have the better performance have a higher chance of being selected, with the difference that using RAM, different chances are given for different individuals depending on their rank which is determined by their fitness. This difference relies on the idea that individuals with worse fitness values are easier to improve than those with better fitness values so they may need different methods. The process would be as follows:

First of all, at the beginning of the generation, the individuals of the population X are sorted and divided into *gr* groups according to their fitness values. The number of individuals in each group, *GS* is calculated with the following expression:

$$GS = \frac{PS}{gr} \in Z^+ \tag{9}$$

Then, each group have M different mutation strategies. The probabilities of all mutation strategies in all groups is represented by P. Therefore, $P_{k,l}$ is the probability of using the *k-th* mutation strategy in the *l-th* group. In order to choose the mutation strategy for an individual, the roulette wheel selection method is used. The pseudocode of this process is shown in the Algorithm 1, where ceil(i/GS) refers to the group id to which the i-th individual belongs and rand(0,1) returns a random value in the range [0,1]. The number of mutation will be 2 as we will see when explaining the integration with JAPDE.

Algorithm 1: function selectMutationStrategy(GS,P)

```
mutationToUse = zeros(ps, 1)
for i = 1 to ps do
    cumulativeP = 0
    k = ceil(i/GS)
    r = rand(0, 1)
    for l = 1 to 2 do
        cumulativeP = cumulativeP + P_{k,l}
        if r ≤ cumulativeP & & mutationToUse(i) == 0 then
            mutationToUse(i) = j
        end
    end
end
```

During a certain period of time called LP_{RAM}, the performance of each mutation strategy for each group is recorded. The probabilities in P are updated according to the performance of the previous learning period. Therefore, the probability of using *l-th* mutation strategy in the *k-th* group ($P_{k,l}$) is calculated with the following expressions:

$$P_{k,l} = (1 - E_{RAM}) \cdot P_{k,l} + E_{RAM} \cdot \Delta P_{k,l} \tag{10}$$

$$\Delta P_{k,l} = \left(\frac{\left(\frac{successP_{k,l}}{triesP_{k,l}}\right)}{\sum_{l=1}^{M}\left(\frac{successP_{k,l}}{triesP_{k,l}}\right)} \right) \tag{11}$$

where E_{RAM} stands for the evaporation rate on RAM which is used to control the impact whenever a probability is changed. $successP_{k_l}$ is the number of times that the *l-th* mutation strategy was used to create a vector with better fitness that its parent in the *k-th* group and $triesP_{k,l}$ is the number of times that the *l-th* mutation strategy was used over a vector of the *k-th* group.

JAPDE [6], on the other side, focus on the adaptation of the parameters F and CR, which often are treat independently. However, as we can see in [6], F and CR are related.

In JAPDE a different value of F and CR are assigned to each individual of the population at the beginning of each generation. These parameters are generated using Cauchy and Normal distributions as shown in the Equations (12) and (13). Where CR_i and F_i are the parameters for the vector x_i. $randomCauchy$ and $randomNormal$ are random generators that use Cauchy and Normal distribution, respectively. Then they are truncated so that they satisfy the conditions (14) and (15). If $F_i < 0$, it is generated again.

$$CR_i = randn_i(\mu_{CR}, 0.05) \tag{12}$$

$$F_i = randc_i(\mu_F, 0.05) \tag{13}$$

$$0 \leq CR_i \leq 1 \tag{14}$$

$$0 \leq F_i \leq 1 \tag{15}$$

In JAPDE a probability distribution matrix (M) is implemented. The size of the matrix M is equal to 11 x 11 where the rows represent different means of CR (from 0 to 1 with steps of 0.1). Therefore, the probability of choosing the *cr-th* mean of CR with the *f-th* mean of F is given by M_{cr_f}.

How a pair of means for F and CR are shown in the algorithm (2), which basically uses a double roulette wheels method in which gets the column (mean of f) and then in that column realizes another roulette to select the row(mean of cr).

After a certain number of generations, called LP_{PA} (learning period), the matrix of probabilities is updated with the following expressions:

$$M_{cr,f} = (1 - E_{PA}) \cdot M_{cr,f} + E_{PA} \cdot \Delta M_{cr,f} \tag{16}$$

$$\Delta M_{cr,f} = \left(\frac{\left(\frac{successM_{cr,f} \cdot EP_f}{triesM_{cr,f}}\right)}{\sum_{cr=0}^{10}\sum_{cr=0}^{10}\left(\frac{successM_{cr,f} \cdot EP_f}{triesM_{cr,f}}\right)} \right) \tag{17}$$

where E_{PA} stands for the evaporation rate on JAPDE, which is used to avoid big changes whenever a probability is modified. $successM_{cr,f}$ is the number of times that a vector generated using the *cr-th* value as mean for parameter CR and the *f-th* value as mean for parameter F had better fitness value than its parent. $triesM_{cr,f}$ is the number of times that the a pair of mean cr and f was used.

EPf is a value from the exploration plane. EP_f is calculated as follows:

$$EP_f = \frac{f}{10} \cdot e^{(\frac{-NFES}{NFES_{MAX}})^3} \tag{18}$$

where $NFES$ is the current number of evaluations and $NFES_{MAX}$ is the maximum number of fitness evaluations. We use 0.01 for f when its value is 0.

The fact that some combinations of meanF and meanCR work well in the early stages of the search but not in the later stages can lead into stagnation problems. That kind of problems gives to the evaporation rate and the exploration plane a reason to be employed, since they are used to avoid fast convergence.

When combining RAM with JAPDE, the following two mutation strategies are used:

Algorithm 2: function meanValuesSelection(M)

meanF $= zeros(ps, 1)$
meanCR $= zeros(ps, 1)$
for $i = 1$ *to ps* **do**
  $probabilityPerF = sumPerRow(M)$
  $r = rand(0, 1)$
  $cumulativeP = probabilityPerF(0)$
  $f = 0$;
  while $cumulativeP < r$ **do**
    $f++$
    $cumulativeP = cumulativeP + probabilityPerF(f)$
  end
  $totalProbabilityF = probabilityPerF(f)$
  $r = rand(0, totalProbabilityF)$
  $cr = 0$
  $cumulativeP = M_{f,cr}$
  while $cumulativeP < r$ **do**
    $cr++$
    $cumulativeP = cumulativeP + M_{f,cr}$
  end
  $meanF(i) = f/10$
  $meanCR(i) = cr/10$
end
return $meanF, meanCR$

- DE/pBest/1

$$v_{i,G} = x_{pBest,G} + F \cdot (x_{r1,G} - x_{r2,G}) \tag{19}$$

- DE/current-to-pbest/1

$$v_{i,G} = X_{i,G} + F_i \cdot (x_{pBest,G} - x_{i,G} + x_{r1,G} - x_{r2,G}) \tag{20}$$

where a parameter called p is used. This parameter will decrease from 1 to $\frac{1}{ps}$ in a deterministic way over the run time. At each generation the value will be given by the following expression:

$$p = max\left(1 - \frac{NFES}{NFES_{MAX}}, \frac{1}{ps}\right) \tag{21}$$

then $x_{pBest,G}$ is a random individual chosen from the top [PS * p]. By using these two mutation strategies and the parameter p we will guarantee that the exploration-exploitation rate will be regulated over the run. As seen in the RAM-JAPDE proposal, when p's value is close to 1, the Eq. 19 behaves as Eq. 1 and when its value is close to 0 it behaves as Eq. 3. The same idea works with Eq 20 which behaves as Eq. 5 when p's value is 1 and like Eq. 6 when its value is close to 0. The pseudocode of RAM-JAPDE is presented in Algorithm 3.

III. Related Work

In this section, an overview of the state-of-the-art algorithms and methods is elaborated. In section **??** , a review of algorithms that adapts different mutation strategies and the control parameters CR and F are given. Then, in section III-B a review of different strategies according to the adaptation of the control parameter PS is done.

Algorithm 3: RAM-JAPDE

// Initialization phase
Initialize M with $1/121$
Initialize P with $1/2$
Calculate GS as in Eq.(9)
Initialize population X
$NFES = 0, G = 0$

// Main loop
while $NFES < NFES_{MAX}$ & $OptimumValueNotFound$ **do**
  Calculate p as in Eq. (21)
  mutationToUse = selectMutationStrategies(GS, P)
  [meanF, meanCR] = meanValuesSelection(M)
  for $i = 1$ *to* ps **do**
    $F_i = Cauchy(meanF(i), 0.05)$
    $CR_i = Noraml(meanCR(i), 0.05)$
    // Mutation
    if *mutationToUse(i) == 1* **then**
      $V_i = X_{pBest} + F_i \cdot (X_{r1} - X_{r2})$
    else
      $V_i = X_i + F_i\dot{(}X_i - X_{pBest} \cdot X_{r1} - X_{r2})$
    end

    //Crossover **for** $j = 1todimension$ **do**
      if $rand(0, 1) < CR_i or j == j_{rand}$ **then**
        $U_i[j] = V_i$
      else
        $U_i[j] = X_i$
      end
    end

    //Selection and counting success and trials
    $f = round(F_i \cdot 10), cr = round(CR_i \cdot 10)$
    $k = ceil(i/GS), l = mutationToUse(i)$
    if $f(U_i) < f(X_i$ **then**
      $X_i = U_i$
      $successM_{cr,f}++$
      $successP_{k,l}++$
    end
    $triesM_{cr,f}++$
    $triesP_{k,l}++$
  end

  //Update M and P
  if *G % LP* **then**
    Update M as in Eq. (16)
    $successM = triesM = 0$
    Update P as in Eq. (10)
    $successP = triesP = 0$
  end
end

A. Strategies that adapt mutation strategies and Parameters CR and F

Depending on the run situation, the use of a mutation operator can result on a better or worse performance compared to the use of the rest of mutation strategies. This is because each mutation strategy has different exploration - exploitation properties. That is why the way of adapt between different mutation strategies during run time becomes an essential task. On the other hand, as we have seen, the performance of the DE algorithm is heavily dependant on the three main parameters (CR, F and PS) which the optimal values can vary according to the nature of the problem and on the stage of the search. Calculate the optimal values at each moment of the search is computationally expensive. In order to adapt these parameters, the state-of-the-art presented some techniques which modify the value of the parameters on the population with deterministic rules, by feedback received from population during the run or by adapting the parameters in each individual on its own feedback.

Some state-of-the-art algorithms that adapt the mutation strategies and the control parameters CR and F are explained below:

- SaDE: [8]: In 2005, A.K.Qin and P.N.Suganthan, presented SaDE, a novel Self adaptive Differential Evolution were two mutation operators (DE/rand/1 and DE/rand-to-best/2 and control parameters F and CR are not required to be pre-specified. In 2009 an improved version added two more mutation operators (DE/rand/2 and DE/current-to-rand/1). At first, an initial probability to be used is given to these four mutation operators. The probabilities are changing over time with the results accumulated over a period time. The more successful mutation operators have more probabilities to be chosen in future mutations. Regarding the control parameters CR and F, both are generated with a Normal Distribution. Nevertheless, CR mean in changing over time with the mean value of the successful trials for each mutation strategy. CR is the only one changing because it was considered more sensitive to problems with different characteristics.
- jDE [9]: In 2006, J. Brest, S. Greiner, B. Bošković, M. Mernik and V. Žumer presented jDE, an algorithm that uses DE/rand/1/bin as the only mutation strategy. In this approach, each individual has his own management of the adaptation of control parameters F and CR. In jDE, the better are the combinations of F and CR, have the higher chances to be used in the creation of future trial vectors.
- JADE [10]: In 2009, J. Zhang and A. C. Sanderson, presented a new algorithm called JADE. In this new approach, a new mutation operator called DE/current-to-pBest/1 was created which incorporated the parameter p. This parameter, which value falls in the range (0,1] was used to select a randomly individual from the top $100 \cdot p\%$ individuals of the population. In JADE, the control parameter CR is generated with a Normal distribution and its mean is updated with the arithmetic mean of the successful CR. The same happens with F, but it is generated with a Cauchy distribution and updated with a Lehmer mean.
- EPSDE [11]: In 2011, R. Mallipedi, P.N. Suganthan, Q.K. Pan and M.F. Tasgetiren presented EPSDE, an algorithm that uses a pool of mutation strategies (each one with diverse characteristics) e.g. DE/best/2, DE/rand/1 and DE/current-to-rand/1 which are used in their comparisons. Regarding to the control parameters CR and F also a pool with combinations of the parameters is used. At the beginning a mutation strategy and a combination of the parameters in given to each individual. If a configuration manages to create a better individual, the combinations are stored into a successful pool for mutation strategies and for combination of parameters. On the contrary, if the configuration fails to produce a better offspring, a new combination is selected from the pool for the next generation.
- CoDE [12]: In 2011, Y. Wang, Z.Cai and Q. Zhang presented an algorithm called CoDE. Its strategy consisted in create a trial vector per each chosen mutation strategy (rand/1/bin, rand/2/bin, current-to-rand/1) with combinations of control parameters ([F=1.0, Cr= 0.1],[F=1.0, Cr= 0.9],[F=0.8, Cr= 0.2]). All the trial vectors compete with other trials and its parent to be kept for the next generation.
- SHADE [13]: In 2013, R. Tanabe and A. Fukunaga presented SHADE with the objective of improving JADE. The algorithm uses the same mutation operator than in JADE, current-to-pbest/1. The difference with JADE is that a different value of p is associated to each individual of the population and it is generated randomly. Regarding to the adaptation of the control parameters, SHADE keeps a record of successful means for F and CR, in this way more reliable values can be generated.
- dn-DADE [14]: In 2014, C. Wang, X. Wang, J. Xiao and Y. Ding presented dn-DADE. This algorithm uses the same approach that SHADE when it comes to mutation strategy (current-to-pbest/1). Nevertheless, in dn-DADE, the parameter p (stated as dn in the proposal) vary with the generations in a nonlinearly manner. When it comes to the control parameters, the values of CR are generated with a Normal distribution but with the novelty that the standard deviation is also updated. Regarding to the scaling factor, F, it is generated with a Cauchy distribution and the mean is adapted during the run in order to have higher mean values at the beginning and smaller mean values at the end of the search.
- sinDE [15]: In 2015 a strategy called sinDE was presented by A. Draa, S. Bouzoubia and I Boukhalfa. This algorithm focused on the adaptation of the control parameters F and CR. In the proposal, six configuration were presented as ways of update F and CR following sinusoidal functions.
- ZEPDE [16]: In 2016, Q. Fan and X. Yan presented ZEPDE. In this strategy, five mutation operators are used. The probability of being applied is updated with the difference between the fitness of the solutions and the worst individual from the new generation. Regarding the control parameters, the algorithm used the concept

of zoning evolution of control parameters. Four regions are established in the space of values for F and CR. Each region has its own center point, which is updated depending on the successful individuals at each region.

B. Adaptation of Control Parameter PS

When it comes to control the population size parameter, two different methods are used [17]. On the one side, we have one kind of method which increases or decreases PS depending on the properties in the population at a certain point. On the other side, we have another kind that modifies PS with a deterministic rule.

When it comes to control the population size parameter, two different methods are used. On the one side, we have one kind that modifies PS with a deterministic rule. On the one side, we have another kind of method which increases or decreases PS depending on the properties in the population at a certain point.

Regarding to the first kind, which seems to be used more, two deterministic methods are used. Both of them reduces PS over time. The difference between them is the way the reduce it.

On the one hand, we have *LPSR* which stands for Linear Population Size Reduction [17]. LSPR is a method proposed by Tanabe and Fukunaga with the idea of presenting L-SHADE [4], the winner of the CEC2014 competition that is based on SHADE. Following the success of the linear method, an extension of the previous algorithm presented as iL-SHADE [18] obtained competitive results. Another successful algorithm based on iL-SHADE called jSO [19] was presented in 2017, which result being the winner of the CEC2017 [20].

In LSPR, a population is being reduced in a linear manner, from an initial population size PS^{init} to a population size PS^{min} at the end of the run. At the end of the generation, the population size required for the next generation is calculated with the following expression:

$$PS_{G+1} = round\left[PS^{init} - \frac{NFES}{NFES_{MAX}} \cdot \left(PS^{init} - PS^{min}\right)\right] \tag{22}$$

When the population size needs to be reduced, the worst $PS_{G+1} - PS_G$ vectors are deleted.

On the other hand, we have *Reduction to the Half approach* presented by J. Brest and M. S. Maucec in the proposition of dyn-NPDE. Similar to LSPR, the population size is reduced from PS^{init} to PS^{min} over the run. Nevertheless, in this approach, when the population size needs to be reduced, we divide the population into halves, then we compare each X_i from the first half. Comparing the elements of the two halves one by one, keeping the individuals with lower fitness value. In the expression (23) we state how the selection of the population from $G+1$ is done from the population G.

$$x_{i,G+1} = \begin{cases} x_{\frac{PS}{2}+i,G} & \text{if} f(x_{\frac{PS}{2}+i,G}) \leq f(x_{i,G}) \\ x_{i,G} & \text{otherwise} \end{cases} \tag{23}$$

with this operator, the population size of the next generation is half of the population size of the previous generation.

The number of different population sizes used in the run is stated as p^{max}. Therefore, the number of population size reductions is $p^{max} - 1$. In the table [I], the quantity of fitness operations and steps can be seen for a PS_{init} of 100 and a p^{max} of 6. The number of fitness operations to be done per each value of PS is calculated as $PS_p \times gen_p$ and the number of generations to do is calculated with the following expression:

$$gen_p = \lfloor \frac{NFES_{MAX}}{p^{max} \cdot PS_p} \rfloor + r_p \tag{24}$$

where r_p is an integer value that represents the number of additional generations to be done when $NFES_{MAX}$ is not multiple of $p^{max} \cdot PS_p$.

TABLE I: Table that displays the amount of generations and fitness evaluations to be done at each PS_p.

p	1	2	3	4	5
PSp	100	50	25	12	6
gen_p	600	1200	2400	5000	10000
$PS_p \times gen_p$	60000	60000	60000	60000	60000

Regarding to the second kind, we can find strategies such as Diversity-Based Mechanism [21]. This mechanism includes a new concept called population diversity, that is used to determinate whether the population has a good balance between exploitation and exploration or not. When using this method, in order to avoid lack of exploitation, if the population is too diverse (exceeds an upper limit), PS is being reduced. On the other side, if the population is not diverse enough(exceeds a lower limit), new individuals will be added to increase the population size and avoid lack of exploration. An example of the successful use of this mechanism is, djSO [22], a variant of djSO that uses diversity-Based Mechanism instead of Linear Population Size Reduction managing to obtain competitive results.

IV. Approaches

In this section, two new algorithms are presented. These are L-RAM-JAPDE and H-RAM-JAPDE. First, an explanation of general considerations when reducing the population size in RAM-JAPDE [6] will be made. Then we will explain the approach for each algorithm.

A. Changes when reducing population

When reducing the population size in RAM-JAPDE, some changes and considerations have to be done. First of all, since PS at some point can not be multiple of gr (amount of groups), the result of applying Eq. (9) does not result in a integer number. In order to fix this situation, we assign each excess individual to different groups. In this way, the maximum amount of individuals between groups is 1.

Another thing to note is that we do not longer have the generations as a counter for our learning periods, since

different generations at different points of the run, will have different ps. Therefore, at early stages learning periods will consist in more fitness evaluations than in later stages of the search. The number of fitness calculations have to be used as a reference instead.

B. L-RAM-JAPDE

In our first approach, Linear Population Size Reduction [4] is incorporated to RAM-JAPDE, which results on L-RAM-JAPDE. The pseudocode for L-RAM-JAPDE can be seen in the Algorithm (4) where the population size control can be seen at the end of the main loop. Initially PS, is set up as 100, which is our PS^{init}. At the end of the loop, the new population size is calculated with the Eq. 22 and the population is reduced if it is required.

Algorithm 4: L-RAM-JAPDE

```
1  // Initialization phase
2  Initialize M with 1/121
3  Initialize P with 1/2
4  Calculate GS as in Eq. (9)
5  Initialize population X
6  NFES = 0, G = 0
7  ps = 100
8
9  // Main loop
10 while NFES < NFES_MAX & OptimumValueNotFound do
11     ...
12     ...
13     Same lines from 10 to 41 as in RAM-JAPDE, Algorithm (3)
14     ...
15     ...
16     numberOfFitEvaOnLP = numberOfFitEvaOnLP + ps
17
18     //Update M and P
19     if numberOfFitEvaOnLP ≥ LPNFES then
20         numberOfFitEvaOnLP = numberOfFitEvaOnLP - LPNFES
21         Update M as in Eq. (16)
22         successM = triesM = 0
23         Update P as in Eq. (10)
24         successP = triesP = 0
25     end
26
27     // Reduce population
28     newPopulationSize = newPopulationSize(NFES)
29     numberOfIndToReduce = ps - newPopulationSize
30     reducePopulation(X, numberOfIndToReduce)
31     ps = newPopulationSize
32 end
```

C. H-RAM-JAPDE

In our second algorithm, Reduction to the Half [23] approach is used, which results on H-RAM-JAPDE. The pseudocode is presented in the Algorithm (5).

At the beginning, PS is initialized with 100 (P^{init}) and p_max is calculated as the number of steps to be reduced, dividing the population until the PS_p can not be divided, because the result would be less than PS_{min}. We also calculated the amount of fitness operations per each PS_p and the number of generations to be done in the first PSp is calculated with the Eq. (24).

At the end of the loop, if we reach the maximum number of generations and we have not done all the reductions, the population is reduced to its half following Eq. (23) and the number of generations to be done in the next value of PS is calculated.

V. Experimental Results

In this section a comparison between the use of two different values of PS is done in first place. Then, a comparison between the new two algorithms proposed with their precursor is done. At the end, L-RAM-JAPDE and H-RAM-JAPDE are compared with some of the state-of-the-art algorithms.

A. Experimental Settings and Comparative measures

In order to compare the performance of the different algorithms in this paper, the benchmark test suite [5] is used, which is composed by the following 30 functions:

- functions from $f1$ to $f3$: are unimodal
- functions from $f4$ to $f16$: are multimodal.
- functions from $f17$ to $f22$: are hybrid.
- functions from $f23$ to $f30$: are composition.

All the results obtained in this paper are the result of running each algorithm 40 times. As indicated in the benchmark instructions, each run was finished after doing 10.000 * Dimension of the problem = 300.000 (30 in our case) fitness function calls or achieving an error below 10^{-8} with respect to the optimal value of the function, which will be considered as 0. In addition, the search space is defined as [-100,100].

In order to compare the algorithms we have used the following measurements:

- Wilconxon signed rank test: A non-parametric statistical test used to compare each algorithms 1 to 1. This test can help to determinate if there are relevant difference between two algorithms. In our case an α = 0.05 is taken in as significant value. If the $p - value$ is smaller than α then it means that the two algorithms are statistically different.
- F.A.R : A measurement calculated dividing the results of each algorithm at each function by the worst performance at each function. Then the sum of each value per each function is taken. The smaller the value the better.
- S.R.E : A measurement calculated by ranking the result of each algorithm at each function. Then the average of all the performances of the algorithm is done. As in F.A.R, the smaller the value the better.

Algorithm 5: H-RAM-JAPDE

```
// Initialization phase
Initialize M with 1/121
Initialize P with 1/2
Calculate GS as in Eq (9)
Initialize population X
NFES = 0, G = 0
ps = 100
pReducionsToDo = calculateReducions(ps)
fitnessOperationsPerP =
  calculateFitnessOperations(pReducionsToDo + 1)
generationsForCurrentP =
  calculateGenerations(fitnessOperationsPerP, ps)

// Main loop
while NFES < NFES_MAX & OptimumValueNotFound do
    ...
    ...
    Same lines from 10 to 41 as in RAM-JAPDE,
     Algorithm [5]
    ...
    ...
    numberOfFitEvaOnLP = numberOfFitEvaOnLP
     + ps

    //Update M and P
    if numberOfFitEvaOnLP ≥ LPNFES then
        numberOfFitEvaOnLP =
         numberOfFitEvaOnLP - LPNFES
        Update M as in Eq. (16)
        successM = triesM = 0
        Update P as in Eq. (10)
        successP = triesP = 0
    end

    // Reduce population
    generationsCounter = generationsCounter ++
    if generationsCounter ≥
     generationsForCurrentP &
     pReducionsToDo ≥ 0 then
        ReducePopulationInHalf(X) ps = ps/2
        generationsCounter = 0
        pReducionsToDo –
        generationsForCurrentP = calculateGenera-
         tions(fitnessOperationsPerP,ps)
    end
end
```

B. Initial Population Size Study

As we have seen in previous sections, one of the parameters to decide is the initial population size (PS_{init}). In the paper where L-Shade is proposed [4], it is indicated that by using ILSParameter, a parameter tunner, 480 was the best PS_{init} for solving the problems with dimension 30. Therefore, a comparison using 100 and 480 as PS_{init} in both of our population size reduction approaches have been done. The configuration for the algorithm is the following: PS_{init} = 100 or 480, PS_{min} = 4, gr = 10, $LP_{PA} = 8000$ fitness evaluations and $E_{RAM} = E_{PA} = 0.2$.

In the tables [II] and [III] a comparison of using a initial population size of 480 against 100 is done.

TABLE II: Comparison between the use of 100 vs 480 as initial value for N_{init} on L-RAM-JAPDE. The results are based on the benchmark from the CEC2014 test suite with dimension 30.

F	L-RAM-JAPDE 100	L-RAM-JAPDE 480
f_1	2.90E+04	**1.10E+03**
f_2	0.00E+00	0.00E+00
f_3	0.00E+00	0.00E+00
f_4	1.28E+01	**9.42E-01**
f_5	2.01E+01	2.02E+01
f_6	**2.36E+00**	3.50E+00
f_7	3.29E-04	**0.00E+00**
f_8	0.00E+00	0.00E+00
f_9	2.69E+01	**2.45E+01**
f_{10}	2.48E+00	**1.64E+00**
f_{11}	1.90E+03	**1.83E+03**
f_{12}	**1.23E-01**	2.28E-01
f_{13}	1.75E-01	**1.68E-01**
f_{14}	2.37E-01	**2.21E-01**
f_{15}	**2.73E+00**	2.76E+00
f_{16}	**9.47E+00**	9.92E+00
f_{17}	7.43E+02	**1.87E+02**
f_{18}	2.28E+01	**9.46E+00**
f_{19}	**2.80E+00**	3.66E+00
f_{20}	9.67E+00	**5.98E+00**
f_{21}	2.10E+02	**9.46E+01**
f_{22}	1.39E+02	**6.65E+01**
f_{23}	3.15E+02	3.15E+02
f_{24}	2.25E+02	**2.24E+02**
f_{25}	2.03E+02	2.03E+02
f_{26}	1.00E+02	1.00E+02
f_{27}	3.60E+02	3.60E+02
f_{28}	8.30E+02	**7.93E+02**
f_{29}	7.63E+02	**7.21E+02**
f_{30}	1.37E+03	**9.19E+02**
F.A.R	1.7	1.3
S.R.E	25.92	20.32
Willconxon	-	1.5111E-03 (diff=yes)

As we can see, for both cases the use of $PS_{init} = 480$ result in better outcome for all functions but for multimodal functions and function f_{19} where using 100 gives an advantage. Since using 480 as initial value for PS outperforms using 100 we will keep the new value for future comparisons.

C. Comparison with RAM-JAPDE

Now that we know a good value for PS_{init} in our population size reduction algorithms, we are going to compare both approaches with their precursor (RAM-JAPDE) [6], in order to check if its performance it is increased. The configuration for this RAM-JAPDE is the following: $PS = 100$, gr = 10, $LP_{RAM} = LP_{PA}$ = 80 generations and $E_{RAM} = E_{PA}$ = 0.2.

On the one hand, in the table [IV] we compare the base algorithm, RAM-JAPDE, with the Linear approach. The results show that the extended version has better performance

TABLE III: Comparison between the use of 100 vs 480 as initial value for N_{init} on H-RAM-JAPDE. The results are based on the benchmark from the CEC2014 test suite with dimension 30.

F	H-RAM-JAPDE 100	H-RAM-JAPDE 480
f_1	1.47E+05	**4.62E+04**
f_2	0.00E+00	0.00E+00
f_3	0.00E+00	0.00E+00
f_4	**4.18E+01**	5.90E+01
f_5	2.01E+01	2.01E+01
f_6	**2.43E+00**	3.62E+00
f_7	0.00E+00	0.00E+00
f_8	3.32E-02	**0.00E+00**
f_9	**2.66E+01**	2.84E+01
f_{10}	3.54E+00	**1.67E+00**
f_{11}	1.93E+03	**1.91E+03**
f_{12}	1.63E-01	**1.62E-01**
f_{13}	1.96E-01	**1.75E-01**
f_{14}	2.38E-01	**2.33E-01**
f_{15}	**2.67E+00**	2.71E+00
f_{16}	**9.58E+00**	9.85E+00
f_{17}	1.08E+03	**3.58E+02**
f_{18}	2.30E+01	**1.62E+01**
f_{19}	3.39E+00	**3.03E+00**
f_{20}	1.03E+01	**6.62E+00**
f_{21}	2.67E+02	**1.30E+02**
f_{22}	1.17E+02	**7.60E+01**
f_{23}	3.15E+02	3.15E+02
f_{24}	2.25E+02	**2.24E+02**
f_{25}	2.04E+02	**2.03E+02**
f_{26}	1.00E+02	1.00E+02
f_{27}	**3.66E+02**	3.74E+02
f_{28}	8.16E+02	**7.95E+02**
f_{29}	9.02E+02	**8.07E+02**
f_{30}	1.43E+03	**1.06E+03**
F.A.R	1.7	1.3
S.R.E	25.26	21.95
Willconxon	-	9.39E-03(yes)

TABLE IV: Comparison between RAM-JAPDE [6] with $PS_{init} = 100$ and L-RAM-JAPDE with $PS_{init} = 480$. The results are based on the benchmark from the CEC2014 test suite with dimension 30.

F	RAM-JAPDE	L-RAM-JAPDE
f_1	1.07E+04	**1.10E+03**
f_2	0.00E+00	0.00E+00
f_3	0.00E+00	0.00E+00
f_4	**2.62E-02**	9.42E-01
f_5	**2.01E+01**	2.02E+01
f_6	**1.55E+00**	3.50E+00
f_7	2.47E-04	**0.00E+00**
f_8	0.00E+00	0.00E+00
f_9	2.49E+01	**2.45E+01**
f_{10}	**1.47E+00**	1.64E+00
f_{11}	**1.73E+03**	1.83E+03
f_{12}	**8.24E-02**	2.28E-01
f_{13}	1.76E-01	**1.68E-01**
f_{14}	2.34E-01	**2.21E-01**
f_{15}	2.80E+00	**2.76E+00**
f_{16}	**9.52E+00**	9.92E+00
f_{17}	4.17E+02	**1.87E+02**
f_{18}	2.02E+01	**9.46E+00**
f_{19}	**3.00E+00**	3.66E+00
f_{20}	1.02E+01	**5.98E+00**
f_{21}	1.45E+02	**9.46E+01**
Ef_{22}	1.24E+02	**6.65E+01**
f_{23}	3.15E+02	3.15E+02
f_{24}	2.24E+02	2.24E+02
f_{25}	2.04E+02	**2.03E+02**
f_{26}	1.00E+02	1.00E+02
f_{27}	3.60E+02	3.60E+02
f_{28}	8.11E+02	**7.93E+02**
f_{29}	**7.17E+02**	7.21E+02
f_{30}	1.36E+03	**9.19E+02**
F.A.R	1.583	1.416
S.R.E	24.44	22.31
Willconxon	-	1.97E-01(no)

on all the functions but the multimodals and the function f_{19}. Even thought it has better results in the F.A.R and S.R.E measurements, it does not manage to pass the Wilconxon test, which means that there are not enough evidence to conclude that the linear version is statistically better.

On the other hand, in the table [V] we compare the base algorithm, RAM-JAPDE, with the Halves reduction method. Similar to the previous results there are no evidence that the extended version is statistically better than the original one. Moreover, the results are even worse, not only some functions are worse in the multimodal but the performance is worse in some other problems outside that category. In addition, S.R.E indicates that the extended is worse too.

D. Comparison with others algorithms of the state of art

Now, we will compare our algorithm with others algorithms that use adaptive methods to change the value of the parameters over run-time. The algorithms and their execution settings used for the comparison are the following ones:

- **L-RAM-JAPDE:** PS_{init} = 480 (18 x n), PS_{min} = 4, gr = 10, LP_{RAM} = LP_{PA} = 8000 fitness evaluations and E_{RAM} = E_{PA} = 0.2
- **H-RAM-JAPDE:** PS_{init} = 480 (18 x n), PS_{min} = 4, p_{max} = 6, gr = 10, LP_{RAM} = LP_{PA} = 8000 fitness evaluations and E_{RAM} = E_{PA} = 0.2
- **CoDE:** PS = 30
- **EPSDE:** PS = 50
- **ZEPDE:** PS = 100, G_{max}= 3000, G_s = 0.175 x G_{max} and G_b= 0.35 x G_{max}
- **SHADE:** PS = 100 and H = 100
- **jDE:** PS = 100, $\tau 1 = \tau 2 = 0.1$, F_l = 0.1 and F_u = 0.9
- **JADE:** PS = 100, A = 100, c = 0.1 and p = 0.1
- **sinDE:** PS = 40, freq = 0.25 and configuration 2

First, in table [VI] a comparison between L-RAM-JAPDE and the previous mentioned algorithms is made. The F.A.R and S.R.E measurements indicate that it is the highest performer of all the algorithms. In addition, the Wilconxon test is performed showing that it is better than CoDE [12], EPSDE [11], ZEPDE [16], jDE [9], JADE [10], sinDE [15]. Nevertheless, observating the Wilconxon test results, it can be seen that L-RAM-JAPDE does not manage to prove that it is statistically better than SHADE [13], which have better results in functions f_1, f_4, f_9 and f_{11}.

In the following table [VII], we have the results of performing a comparison between H-RAM-JAPDE and its coun-

TABLE V: Comparison between RAM-JAPDE [6] with $PS_{init} = 100$ and H-RAM-JAPDE with $PS_{init} = 480$. The results are based on the benchmark from the CEC2014 test suite with dimension 30.

F	RAM-JAPDE	H-RAM-JAPDE
f_1	**1.07E+04**	4.62E+04
f_2	0.00E+00	0.00E+00
f_3	0.00E+00	0.00E+00
f_4	**2.62E-02**	5.90E+01
f_5	2.01E+01	2.01E+01
f_6	**1.55E+00**	3.62E+00
f_7	2.47E-04	**0.00E+00**
f_8	0.00E+00	0.00E+00
f_9	**2.49E+01**	2.84E+01
f_{10}	**1.47E+00**	1.67E+00
f_{11}	**1.73E+03**	1.91E+03
f_{12}	**8.24E-02**	1.62E-01
f_{13}	1.76E-01	**1.75E-01**
f_{14}	2.34E-01	**2.33E-01**
f_{15}	2.80E+00	**2.71E+00**
f_{16}	**9.52E+00**	9.85E+00
f_{17}	4.17E+02	**3.58E+02**
f_{18}	2.02E+01	**1.62E+01**
f_{19}	**3.00E+00**	3.03E+00
f_{20}	1.02E+01	**6.62E+00**
f_{21}	1.45E+02	**1.30E+02**
f_{22}	1.24E+02	**7.60E+01**
f_{23}	3.15E+02	3.15E+02
f_{24}	2.24E+02	2.24E+02
f_{25}	2.04E+02	**2.03E+02**
f_{26}	1.00E+02	1.00E+02
f_{27}	**3.60E+02**	3.74E+02
f_{28}	8.11E+02	**7.95E+02**
f_{29}	**7.17E+02**	8.07E+02
f_{30}	1.36E+03	**1.06E+03**
F.A.R	1.52	1.48
S.R.E	23.64	24.53
Wilconxon	-	9.3E-01(no)

terparts. Similar to L-RAM-JAPDE, H-RAM-JAPDE obtains the best results in most of the functions. It also obtains good scores in F.A.R and S.R.E measurements, only outperformed by SinDE in S.R.E. Regarding to the Wilconxon test, it can be seen that the new method is statistically better than CODE, EPSDE, ZEPDE and jDE. Moreover, the results show that H-RAM-JAPDE is not statistically better than SHADE, JADE or SinDE.

VI. Conclusion

Avoiding slow convergence and stagnation into a local optima is heavily required in order to guarantee a good performance of the differential evolution algorithm. One way of achieving it is by controlling the parameters F, CR and PS. The purpose of this work is to show how modifying the third parameter helps to avoid these issues and provides a good exploration and exploitation factor. These modifications were conducted on the algorithm RAM-JAPDE resulting in two algorithms, one per each approach, L-RAM-JAPDE which uses lineal reduction and H-RAM-JAPDE which uses reduction to the half.

In order to carry out the comparison, the benchmark proposed in the CEC2014 conference was used. The results shown that using 480 as initial population size instead of 100 was coming off better outcomes. From the results, we can also see that even though L-RAM-JAPDE was not statistically better than RAM-JAPDE, it was outperforming in almost all of the functions compared to RAM-JAPDE (outside of multimodal functions). In addition, RAM-JAPDE was, except for SHADE, statistically better than all the state-of-the-art presented in this paper. On the other side, H-RAM-JAPDE, even though it obtained good results at some measurements, it fails on proving that is was statistically better than RAM-JAPDE and some other algorithms. From the results we can conclude that the new algorithms were performing significantly better almost at all the functions outside of the ones of multimodal nature. This means that is worth trying to reduce PS over time at least in a linear manner, when trying to deal with problems that consider unimodal, hybrid or composition functions.

VII. Future Work

Future works will focus on applying methods, that dynamically modify the control parameter PS during the run, in order to extend other DE variants and try if that kind of approach enhance their performance. Another works could contemplate the study of different initial population sizes, which could result in an adequate exploration, and minimum population size, which could result in better exploitation. It would also be interesting to compare the results between applying a reduction method, like the ones used in this paper, and a method that can increase or decrease PS regarding to the feedback from the run.

References

[1] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[2] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[3] P. N. S. R Mallipeddi, "Empirical study on the effect of population size on differential evolution algorithm," *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, 2008.

[4] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, 2014, pp. 1658–1665.

[5] J. Liang and P. N. Suganthan, "Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization," *IEEE Transactions on Cybernetics*, 2013.

[6] X. N. Leon M, "Adaptive differential evolution with a new joint parameter adaptation method," *Journal of global optimization*, 2019.

[7] L. M and X. N, "Enhancing adaptive differential evolution algorithms with rank-based mutation adaptation," in *2018 IEEE congress on evolutionary computation, CEC 2018—proceedings*. IEEE, 2018, pp. 1–7.

[8] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[9] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[10] J. Zhang and A. C. Sanderson, "Jade: Adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

TABLE VI: Comparison between L-RAM-JAPDE with PS_{init} = 480, CoDE [12], EPSDE [11], ZEPDE [16], SHADE [13], jDE [9], JADE [10] and sinDE [15]. The results are based on the benchmark from the CEC2014 test suite with dimension 30.

F	L-RAM-JAPDE	CoDE	EPSDE	ZEPDE	SHADE	jDE	JADE	sinDE
f_1	1.10E+03	9.68E+04	4.18E+06	1.31E+06	**2.87E+02**	6.18E+05	7.55E+02	8.56E+05
f_2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_3	0.00E+00	0.00E+00	0.00E+00	3.25E-08	0.00E+00	0.00E+00	**5.41E-04**	0.00E+00
f_4	9.42E-01	**2.57E-01**	5.71E-01	8.02E+00	0.00E+00	1.84E+00	0.00E+00	7.73E-01
f_5	2.02E+01	2.01E+01	2.09E+01	2.08E+01	2.02E+01	2.06E+01	2.03E+01	2.03E+01
f_6	3.50E+00	3.37E+00	7.39E-01	1.12E+00	9.98E+00	1.67E-01	9.98E+00	9.02E-02
f_7	0.00E+00	1.48E-03	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_8	0.00E+00	1.53E+01	4.69E+01	1.51E+01	0.00E+00	2.40E-02	0.00E+00	6.57E+00
f_9	2.45E+01	7.40E+01	1.51E+02	3.60E+01	**2.41E+01**	1.07E+02	4.59E+01	2.48E+01
f_{10}	**1.64E+00**	1.93E+02	1.94E+03	1.44E+02	2.78E-03	2.11E+02	6.94E-03	9.99E+01
f_{11}	2.05E+03	3.27E+03	6.20E+03	3.63E+03	**1.63E+03**	4.59E+03	2.64E+03	1.70E+03
f_{12}	3.06E-01	**1.28E-01**	1.94E+00	1.35E+00	2.15E-01	9.26E-01	3.64E-01	1.78E-01
f_{13}	**1.68E-01**	4.39E-01	2.72E-01	2.04E-01	2.31E-01	3.22E-01	3.11E+00	1.73E-01
f_{14}	**2.21E-01**	3.41E-01	2.60E-01	2.39E-01	2.50E-01	2.73E-01	2.41E+00	2.26E-01
f_{15}	**2.76E+00**	1.09E+01	1.37E+01	6.29E+00	2.81E+00	1.06E+01	4.18E+00	3.50E+00
f_{16}	9.92E+00	1.14E+01	1.22E+01	1.15E+01	9.46E+00	1.13E+01	9.32E+00	**8.80E+00**
f_{17}	**1.87E+02**	5.30E+03	3.56E+05	6.18E+04	9.65E+02	2.48E+03	1.23E+03	5.18E+03
f_{18}	**9.46E+00**	6.67E+01	4.42E+02	2.81E+02	5.77E+01	4.33E+01	7.79E+01	1.74E+01
f_{19}	3.66E+00	4.77E+00	5.09E+00	5.15E+00	4.67E+00	4.95E+00	4.42E+00	**2.55E+00**
f_{20}	**5.98E+00**	3.70E+01	5.88E+01	6.64E+01	1.48E+01	2.15E+01	2.86E+03	1.09E+01
f_{21}	**9.46E+01**	1.23E+03	8.50E+03	5.50E+03	2.45E+02	5.54E+02	1.58E+04	3.59E+02
f_{22}	6.65E+01	3.85E+02	1.38E+02	5.62E+01	1.30E+02	4.97E+01	1.64E+02	9.91E+01
f_{23}	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02
f_{24}	2.24E+02	2.27E+02	2.26E+02	2.24E+02	2.27E+02	**2.23E+02**	2.26E+02	2.24E+02
f_{25}	2.03E+02	2.04E+02	2.05E+02	2.04E+02	2.03E+02	2.03E+02	2.04E+02	2.03E+02
f_{26}	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.04E+02	1.00E+02	1.00E+02	1.00E+02
f_{27}	3.60E+02	3.79E+02	3.15E+02	**3.02E+02**	3.35E+02	3.45E+02	3.50E+02	3.07E+02
f_{28}	7.93E+02	8.57E+02	8.17E+02	8.08E+02	8.17E+02	7.91E+02	**7.85E+02**	8.02E+02
f_{29}	7.21E+02	8.80E+02	2.41E+03	1.98E+03	7.21E+02	9.10E+02	7.29E+02	9.15E+02
f_{30}	**9.19E+02**	1.57E+03	1.73E+03	1.66E+03	1.77E+03	1.08E+03	1.53E+03	1.32E+03
F.A.R	2.80	5.40	6.32	5.37	3.60	4.5	4.73	3.28
S.R.E	10.70	15.27	20.96	15.67	12.18	13.32	17.38	10.77
Wilconxon	-	8.17E-07(yes)	1.70E-05(yes)	1.41E-04(yes)	4.98E-01(no)	4.40E-03(yes)	9.00E-03(yes)	3.20E-02(yes)

[11] Q. K. P. R Mallipeddi, P N Suganthan and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Journal of global optimization*, vol. 11, no. 4, pp. 1679–1696, 2011.

[12] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, 2011.

[13] A. F. R. Tanabe, "Success-history based parameter adaptation for differential evolution," *Proc.2013 IEEE Congress on Evolutionary Computation (CEC)*, pp. 71–78, 2013.

[14] J. X. C Wang, X Wang and Y. Ding, "Improved differential evolution algorithm based on dynamic adaptive strategies and control parameters," *International Journal of Control and Automation*, vol. 7(9), pp. 81,96.

[15] S. B. A Draa and I. Boukhalfa, *A sinusoidal differential evolution algorithm for numerical optimisation, Applied Soft Computing*, 2015, vol. 27.

[16] Q. Fan and X. Yan, "Self-adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies," *IEEE Transactions on Cybernetics*, vol. 46, no. 1, pp. 219–232, 2016.

[17] P. B. R Poláková, "Adaptation of population size in differential evolution algorithm: An experimental comparison," *25th International Conference on Systems, Signals and Image Processing(IWSSIP)*, 2018.

[18] J. Brest, M. S. Maučec, and B. Bošković, "il-shade: Improved l-shade algorithm for single objective real-parameter optimization," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 1188–1195.

[19] ——, "Single objective real-parameter optimization: Algorithm jso," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2017, pp. 1311–1318.

[20] B. Y. Q. M Z Ali, J J Liang and P. N. Suganthan, "Definitions and evaluation criteria for the cec 2017 special session and competition on single objective real-parameter numerical optimization," *Nanyang Technological University, Jordan University of Science and Technology and Zhengzhou University*, 2016.

[21] J. T. R Poláková and P. Bujok, "Adaptation of population size according to current population diversity in differential evolution," *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017.

[22] R. Poláková, J. Tvrdík, and P. Bujok, "Adaptation of population size according to current population diversity in differential evolution," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017, pp. 1–8.

[23] J. Brest and M. S. Maučec, "Population size reduction for the differential evolution algorithm," *Applied Intelligence, The International Journal of Research on Intelligent Systems for Real Life Complex Problems*, vol. 29, pp. 228–247, 2007.

TABLE VII: Comparison between L-RAM-JAPDE with N_{init} = 480, CoDE [12], EPSDE [11], ZEPDE [16], SHADE [13], jDE [9], JADE [10] and sinDE [15]. The results are based on the benchmark from the CEC2014 test suite with dimension 30.

F	H-RAM-JAPDE	CODE	EPSDE	ZEPDE	SHADE	jDE	JADE	sinDE
f_1	4.62E+04	9.68E+04	4.18E+06	1.31E+06	2.87E+02	6.18E+05	7.55E+02	8.56E+05
f_2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_3	0.00E+00	0.00E+00	0.00E+00	3.25E-08	0.00E+00	0.00E+00	**5.41E-04**	0.00E+00
f_4	5.90E+01	**2.57E-01**	5.71-01	8.02E+00	0.00E+00	1.84E+00	0.00E+00	7.73E-01
f_5	2.01E+01	2.01E+01	2.09E+01	2.08E+01	2.02E+01	2.06E+01	2.03E+01	2.03E+01
f_6	3.62E+00	3.37E+00	7.39E-01	1.12E+00	9.98E+00	1.67E-01	9.98E+00	**9.02E-02**
f_7	0.00E+00	**1.48E-03**	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_8	0.00E+00	1.53E+01	4.69E+01	1.51E+01	0.00E+00	**2.40E-02**	0.00E+00	6.57E+00
f_9	2.84E+01	7.40E+01	1.51E+02	3.60E+01	**2.41E+01**	1.07E+02	4.59E+01	2.48E+01
f_{10}	**1.67E+00**	1.93E+02	1.94E+03	1.44E+02	2.78E-03	2.11E+02	6.94E-03	9.99E+01
f_{11}	1.91E+03	3.27E+03	6.20E+03	3.63E+03	**1.63E+03**	4.59E+03	2.64E+03	1.70E+03
f_{12}	1.62E-01	**1.28E-01**	1.94E+00	1.35E+00	2.15E-01	9.26E-01	3.64E-01	1.78E-01
f_{13}	1.75E-01	4.39E-01	2.72E-01	2.04E-01	2.31E-01	3.22E-01	3.11E+00	**1.73E-01**
f_{14}	2.33E-01	3.41E-01	2.60E-01	2.39E-01	2.50E-01	2.73E-01	2.41E+00	**2.26E-01**
f_{15}	**2.71E+00**	1.09E+01	1.37E+01	6.29E+00	2.81E+00	1.06E+01	4.18E+00	3.50E+00
f_{16}	9.85E+00	1.14E+01	1.22E+01	1.15E+01	9.46E+00	1.13E+01	9.32E+00	**8.80E+00**
f_{17}	**3.58E+02**	5.30E+03	3.56E+05	6.18E+04	9.65E+02	2.48E+03	1.23E+03	5.18E+03
f_{18}	**1.62E+01**	6.67E+01	4.42E+02	2.81E+02	5.77E+01	4.33E+01	7.79E+01	1.74E+01
f_{19}	3.03E+00	4.77E+00	5.09E+00	5.15E+00	4.67E+00	4.95E+00	4.42E+00	**2.55E+00**
f_{20}	**6.62E+00**	3.70E+01	5.88E+01	6.64E+01	1.48E+01	2.15E+01	2.86E+03	1.09E+01
f_{21}	**1.30E+02**	1.23E+03	8.50E+03	5.50E+03	2.45E+02	5.54E+02	1.58E+04	3.59E+02
f_{22}	7.60E+01	3.85E+02	1.38E+02	5.62E+01	1.30E+02	**4.97E+01**	1.64E+02	9.91E+01
f_{23}	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02
f_{24}	2.24E+02	2.27E+02	2.26E+02	2.24E+02	2.27E+02	**2.23E+02**	2.26E+02	2.24E+02
f_{25}	2.03E+02	2.04E+02	2.05E+02	2.04E+02	2.03E+02	2.03E+02	2.04E+02	2.03E+02
f_{26}	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.04E+02	1.00E+02	1.00E+02	1.00E+02
f_{27}	3.74E+02	3.79E+02	3.15E+02	**3.02E+02**	3.35E+02	3.45E+02	3.50E+02	3.07E+02
f_{28}	7.95E+02	8.57E+02	8.17E+02	8.08E+02	8.17E+02	7.91E+02	**7.85E+02**	8.02E+02
f_{29}	8.07E+02	8.80E+02	2.41E+03	1.98E+03	**7.21E+02**	9.10E+02	7.29E+02	9.15E+02
f_{30}	**1.06E+03**	1.57E+03	1.73E+03	1.66E+03	1.77E+03	1.08E+03	1.53E+03	1.32E+03
F.A.R	2.98	5.42	6.32	5.33	3.63	4.43	4.67	3.22
S.R.E	11.69	15.24	20.90	14.81	12.18	13.13	17.38	10.69
Willconxon	-	6.84E-06(yes)	1.67E-04(yes)	1.88E-03(yes)	9.00E-01(no)	2.20E-02(yes)	8.70E-02(no)	1.01E-01(no)