COLSHADE for Real-World Single-Objective Constrained Optimization Problems

Javier Gurrola-Ramos
Department of Computer Science
Mathematics Research Center
Guanajuato, Mexico
francisco.gurrola@cimat.mx

Arturo Hernández-Aguirre
Department of Computer Science
Mathematics Research Center
Guanajuato, Mexico
artha@cimat.mx

Oscar Dalmau-Cedeño
Department of Computer Science
Mathematics Research Center
Guanajuato, Mexico
dalmau@cimat.mx

***Abstract*—In this paper we present the COLSHADE algorithm for real parameter constrained optimization problems. COLSHADE evolved from the basic L-SHADE algorithm by introducing significant features such as adaptive Lévy flights and dynamic tolerance (included in the constraint handling technique). Lévy flights mainly perform the exploration phase in algorithms such as the Firefly algorithm and Cuckoo search; in COLSHADE, however, the goal of the Lévy flights is to administer the selection pressure exerted over the population as to find the feasible region and keeping diversity. Thus, a new adaptive Lévy flight mutation operator is introduced here and called the levy/1/bin. In many problems the levy/1/bin excels during the exploration phase whilst the exploitation phase is performed by current-to-pbest mutation. However, the adaptive strategy propitiates the emergence of these two mutation approaches at different rates and times. The proposed method is tested on 57 constrained optimization functions of the benchmark provided for the CEC 2020 real-world single-objective constrained optimization competition.**

***Index Terms*—Constrained optimization, Lévy flight, parameter adaptation.**

I. Introduction

In real life many design problems belong to the constrained optimization category, same which can be formally described as follows:

$$\begin{array}{lll} \text{minimize} & f(\boldsymbol{x}), & \boldsymbol{x} = (x_1 \ldots, x_D) \in \Omega \\ \text{subject to:} & g_i(\boldsymbol{x}) \leq 0, & i = 1, \ldots p \\ & h_j(\boldsymbol{x}) = 0, & j = p+1, \ldots, m \end{array} \quad (1)$$

where $f(\boldsymbol{x})$ is the objective function. $\boldsymbol{x} \in \Omega$ is a D dimensional solution vector and x_i is the i-th component of $\boldsymbol{x}$. $\Omega = \prod_{i=1}^{D}[L_i, U_i]$ is the cartesian product defining the search space and L_i, U_i are the lower bound and the upper bound of x_i, respectively. The function $g_i(\boldsymbol{x})$ is the i-th inequality constraint and $h_j(\boldsymbol{x})$ is the j-th equality constraint. Either constraint can be linear or not linear. Usually equality constraints are transformed into inequalities of the form

$$|h_j(\boldsymbol{x})| - \epsilon \leq 0, \quad \text{for } j = p+1, \ldots, m \quad (2)$$

In this paper we describe a new algorithm, called "L-SHADE for Constrained Optimization with Lévy Flights", COLSHADE, that is applicable to constrained optimization problems of the kind defined above. Therefore, COLSHADE was born with CR and F parameters adaptation, a mutation operator named DE/current-to-$p$best, and the linearly decreasing population feature of its predecessor. COLSHADE incorporates the following new features: a) a new mutation operator levy/1/bin based on the Lévy distribution; b) the adaptation of the Lévy distribution parameter that controls the extension of the step size, therefore, a mixture of short and large mutations that conforms Lévy flights is randomly generated but their extent is tuned up in concordance with successful mutations in the search space; c) a constraint handling technique with dynamic tolerance for equality constraints.

Lévy flights have been used in global optimization to perform the exploratory step of the algorithm, for instance, Cuckoo search [1] and Firefly [2] algorithms. Lévy flight is adopted by our proposed algorithm to keep exploration and sustain the population diversity required to counter balance the negative effect of the selection pressure. In that trade off between exploration and keeping feasible individuals, the selection pressure that favors feasible individuals is kept constant (rules do not change), however, exploration may be sustained along large number of generations since the ultimate goal of the adaptive Lévy flight is the success of the exploration (i.e., finding better fitness individuals).

The rest of this paper is organized as follows. Section II introduces constraint handling, Lévy flights and for the sake of completeness a brief review of L-SHADE algorithm. In Section III our proposal COLSHADE algorithm is presented. Specification of experiments and control parameter setting are given in Section IV. Section V provides experimental results and the algorithm complexity. Finally, Section VI provides final remarks about this work.

II. Background and Related Work

A. Handling Constraints

The adopted constraint handling is based on feasibility rules which implements a constraint handling technique called "separation of constraints and objective" [3]. These rules are quite greedy since given a pair of individuals the winner is the individual with "less total amount of constraints violation", or the one that is feasible if the other is not feasible, or given two feasible individuals the one with better objective function value is chosen. There is no opportunity of survival for the weakest individual in the feasibility sense. Nonetheless, it makes sense

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

to leave some of them alive as a measure to keep diversity and exploration.

B. Lévy Flight

Lévy flights (LF) are random walks with step lengths simulated from a heavy tail distribution, such as the Lévy probability distribution. A visual inspection of LF in two dimensions shows clusters of many small steps linked by sporadic steps of larger size. The clusters of regions randomly spread over the search space where either region is traversed with many small steps is the attractive feature currently exploited by bio-inspired algorithms to add or enhance their exploratory capacity.

The foraging behaviour of some species in their natural environment is being studied since their motion can be explained by LF [4].

Heavy tail distributions such as Cauchy and Lévy probability density functions have already been used in evolutionary algorithms See Fig. 1. Yao et. al. [5], [6] investigated heavy tail distributions as primary search operators in evolutionary programming. The infinite variance of the Cauchy distribution, and the mentioned features of Lévy flights improved the search capacity of the mutation operator. Therefore, new individuals visited regions located farther from their parents in the landscape.

The mathematical model of the Lévy distribution is a particular case of stable distribution S with four parameters (see [7], and Fig. 1): 1) α controls the shape; 2) β controls the skewness of the distribution, that is, positive tail ($\beta > 0$) , negative tail ($\beta < 0$), or symmetric ($\beta = 0$); 3) γ is the scale factor, that is, it controls the step size of a Lévy flight, it is an adaptable parameter in our proposed algorithm; 4) δ is the mean of the distribution.

Recently, Lévy flights were *reintroduced* to the evolutionary computation community with the Cuckoo Search algorithm (CS) [1]. In the CS metaphor, the cuckoo bird randomly flies to a new place to nest, staying in that spot if the fitness is better than in the previous spot. The random motion of the cuckoo bird is modelled after a Lévy distribution, in fact, the new spot is one step of a Lévy flight. LF were incorporated into the Gray Wolf optimization algorithm to redistribute wolves around the fitness landscape, therefore, preventing loss of diversity and stagnation [8]. The Lévy Firefly algorithm for global optimization [2], uses LF to model the random motion of fireflies.

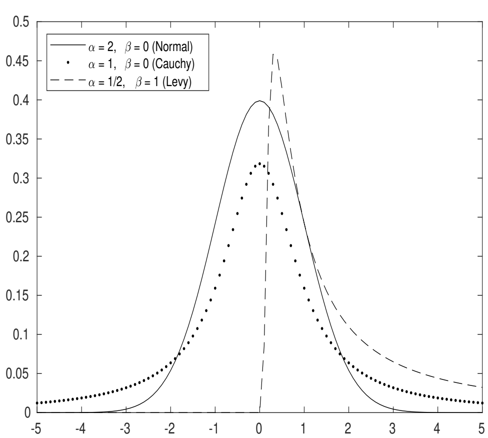

Fig. 1. Normal, Cauchy, and Lévy density functions. Note that either density is generated with a specific value of the α end β parameters.

C. L-SHADE Algorithm

The L-SHADE algorithm [9] is an improvement of Differential Evolution [10]. The current-to-pbest/1/bin strategy proposed by JADE algorithm [11] and defined in equation 3 is used by SHADE [12] and also adopted by L-SHADE algorithm.

$$\boldsymbol{v}_{i,g} = \boldsymbol{x}_{i,G} + F_i \cdot (\boldsymbol{x}_{pbest,G} - \boldsymbol{x}_{i,G}) + F_i \cdot (\boldsymbol{x}_{r1,G} - \boldsymbol{x}_{r2,G}) \quad (3)$$

where $\boldsymbol{x}_{pbest,G}$ is randomly selected among the p-best solutions.

L-SHADE mantains a historical memory M_CR and M_F of H entries for control parameters CR and F respectively. In the beginning, the content of memories is initialized to 0.5. In each generation, the control parameters CR_i and F_i used by each individual are generated selecting a random index r_i in range $[1, H]$ and applying the following criterion:

$$CR_i = \begin{cases} \text{randn}_i(M_{CR,ri}, 0.1) & \text{if } M_{CR,ri} > 0 \\ 0 & \text{other case} \end{cases} \quad (4)$$

$$F_i = \text{randc}_i(M_{F,ri}, 0.1) \quad (5)$$

where $\text{randn}_i(\mu, \sigma)$ and $\text{randc}_i(\delta, \gamma)$ are normal and Cauchy random number generators respectively with mean μ, standard deviation σ, center δ and scale factor γ. The L-SHADE algorithm incorporates linear population size reduction. The algorithm starts with a large population to encourage wide exploration of the search space, and slowly decreases the population size along the generations to accelerate convergence and exploit the best solutions found.

III. COLSHADE Algorithm for Constrained Optimization

In this section, the proposed COLSHADE algorithm is presented. The whole algorithm is listed in Algorithm 1. The aim of the adaptive Lévy flight-based mutation is to achieve larger exploration of the search space, whereas the current-to-$p$best mutation complements the optimization process as an exploitation operator.

A. Lévy Flight-based Mutation

This mutation is listed in Algorithm 2. The magnitude of the Lévy flight and crossover rate CR are adapted throughout the evolutionary process. In that way larger or shorter flights can be generated. The trial vectors are generated starting from

Algorithm 1 COLSHADE

$N^{init} = \text{round}(D \times r^{N^{init}})$, $|\boldsymbol{A}| = \text{round}(N^{init} \times r^{arc})$;
$N_0 = N^{init}$, $N_{min} = 4$, $q_0 = 0.5$;
Initialize memories M_{CR}, M_F, M_{CR_L}, M_{F_L} to 0.5;
Initialize population $P_0 = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_0}\}$;
Set initial tolerance such that all equality constraints are feasible: $\epsilon_{0,j} = \max\{|h_j(\boldsymbol{x}_i)|\}$;
while The termination criteria does not meet **do**
  $S_{CR} \leftarrow \emptyset$, $S_F \leftarrow \emptyset$, $S_{CR_L} \leftarrow \emptyset$, $S_{F_L} \leftarrow \emptyset$;
  $\Delta f \leftarrow \emptyset$, $\Delta f_L \leftarrow \emptyset$;
  for $i = 1$ to N_G **do**
    $l \leftarrow \text{rand}(0, 1)$;
    if $l \leq q_G$ **then**
      CR_i, $F_i \leftarrow \text{GenerateParameters}(M_{CR_L}, M_{F_L})$.
      $\boldsymbol{u}_{i,G} \leftarrow \text{current-to-}p\text{best/1/bin}(\boldsymbol{x}_i, CR_i, F_i, P_G, pbest)$;
    else
      CR_i, $F_i \leftarrow \text{GenerateParameters}(M_{CR}, M_F)$.
      $\boldsymbol{u}_{i,G} \leftarrow \text{levy/1/bin}(\boldsymbol{x}_i, CR_i, F_i, P_G, pbest)$;
    end if
    if $\boldsymbol{u}_{i,G}$ improves $\boldsymbol{x}_{i,G}$ **then**
      $\boldsymbol{x}_{i,G+1} = \boldsymbol{u}_{i,G}$;
      Copy $\boldsymbol{x}_{i,G}$ to archive $\mathbf{A}$;
      if $l \leq q_G$ **then**
        $CR_i \rightarrow S_{CR_L}$, $F_i \rightarrow S_{F_L}$;
        Update Δf_L according to improve criteria;
      else
        $CR_i \rightarrow S_{CR}$, $F_i \rightarrow S_F$;
        Update Δf according to improve criteria;
      end if
    end if
  end for
  $M_{CR}, M_F \leftarrow \text{UpdateMemories}(S_{CR}, S_F, \Delta f)$;
  $M_{CR_L}, M_{F_L} \leftarrow \text{UpdateMemories}(S_{CR_L}, S_{F_L}, \Delta f_L)$;
  $q_{G+1} \leftarrow \text{UpdateProbability}(\Delta f, \Delta f_L, q_G, \mu)$;
  $\epsilon_{G+1} \leftarrow \text{UpdateTolerance}(P_G, FEs, FEs_\epsilon, \epsilon_G, \epsilon_f)$;
  $N_{G+1} = \text{round}\left[\left(\frac{N^{min} - N^{init}}{MAX_{FEs}}\right) \times FEs + N^{init}\right]$;
  Resize P_{G+1} according to N_{G+1} deleting $N_G - N_{G+1}$ worst individuals;
  If necessary resize $\boldsymbol{A}$ according to $|P_{G+1}|$ deleting randomly individuals in $\boldsymbol{A}$;
end while

a solution x_i in direction to x_{pbest}. In contrast to current-to-$p$best, all the variables of the trial vector have different scale factor $F_j \in (F_{crit}, 1)$ in order to increase exploration in a promising direction. The value F_{crit} [13] limits the minimum value of F given CR in order to avoid the rapid loss of diversity in the population.

B. Constraint Handling

Objective function and constraints are handled separately. Feasibility is measured by the sum of constraints violation:

Algorithm 2 levy/1/bin mutation

Ensure: New individual $\boldsymbol{u}$;
$F_{crit} \leftarrow \sqrt{(1 - CR_i/2)/N_G}$.
$F_i \leftarrow \max\{F_{cri}, F_i\}$.
$j_{rand} \leftarrow \text{RandInteger}(1, D)$.
Choose randomly $\boldsymbol{x}_{pbest}$ from P_G.
for $j = 1$ to D **do**
  if rand(0, 1) $\leq CR$ or $j = j_{rand}$ **then**
    $F_{levy} \leftarrow F_i \cdot S(\alpha, \beta, \gamma, \gamma + \delta)$.
    $u_j \leftarrow x_j + F_{levy} \cdot (x_{pbest,j} - x_j)$.
  else
    $u_j \leftarrow x_j$.
  end if
end for

$$svc(\boldsymbol{x}, \boldsymbol{\epsilon}_G) = \mathcal{I}(\boldsymbol{x}) + \mathcal{E}(\boldsymbol{x}, \boldsymbol{\epsilon}_G) \tag{6}$$

$$\mathcal{I}(\boldsymbol{x}) = \sum_{i=1}^{p} \max\{g_i(\boldsymbol{x}), 0\} \tag{7}$$

$$\mathcal{E}(\boldsymbol{x}, \boldsymbol{\epsilon}_G) = \sum_{j=p+1}^{m} \max\{|h_j(\boldsymbol{x})| - \epsilon_{j,G}, 0\} \tag{8}$$

where $\boldsymbol{\epsilon}_G$ is the tolerance vector for equality constraints. Each constraint $h_j(x)$ is associated with a component $\epsilon_{j,G}$ of the tolerance vector. In each generation the tolerance $\boldsymbol{\epsilon}_G$ is adapted dynamically using the following exponential strategy:

$$\epsilon_{j,G+1} = \begin{cases} \epsilon_{j,G} \times \left(\frac{\epsilon_f}{\epsilon_{j,G}}\right)^{\frac{1}{FEs_\epsilon - FEs}} & \text{If } p_G \text{ individuals are } \epsilon\text{-feasible.} \\ \epsilon_{j,G} & \text{Other case.} \end{cases} \tag{9}$$

where ϵ_f is the final tolerance, FEs_ϵ is the maximum number of fitness evaluations allowed with a tolerance $\epsilon_{j,G} > \epsilon_f$ and $p_G \leq P_G$ is the number of current feasible individuals necessary for update the tolerance.

The feasibility binary tournament selection based on feasibility rules [3] picks individuals for recombination. To encourage the search of the feasible region, the case when two feasible individuals are compared (for some $\epsilon-$tolerance), the one closer to the feasible region is preferred. The feasibility binary tournament is defined as follows:

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_i & svc(\boldsymbol{u}_i; \boldsymbol{\epsilon}_G) < svc(\boldsymbol{x}_{i,G}; \boldsymbol{\epsilon}_G). \\ \boldsymbol{u}_i & svc(\boldsymbol{u}_i; \boldsymbol{\epsilon}_G) = svc(\boldsymbol{x}_{i,G}; \boldsymbol{\epsilon}_G) = 0 \\ & \wedge f(\boldsymbol{u}_i) < f(\boldsymbol{x}_{i,G}). \\ \boldsymbol{u}_i & svc(\boldsymbol{u}_i; \boldsymbol{\epsilon}_G) = svc(\boldsymbol{x}_{i,G}; \boldsymbol{\epsilon}_G) = 0 \\ & \wedge svc(\boldsymbol{u}_i; \boldsymbol{\epsilon}_f) < svc(\boldsymbol{x}_{i,G}; \boldsymbol{\epsilon}_f). \\ \boldsymbol{x}_{i,G} & \text{other case.} \end{cases} \tag{10}$$

The first and third cases of the tournament use the feasibility of the individuals as the selection criteria, while the second case considers optimality as selection criteria. Note that the second criteria allows to select individuals such that $svc(\boldsymbol{x}_{i,G};\boldsymbol{\epsilon}_f) < svc(\boldsymbol{u}_i;\boldsymbol{\epsilon}_f)$, this allows exploring the not feasible - feasible boundary when $\epsilon_{j,G} \rightarrow \epsilon_f$, avoiding to be left isolated in suboptimal regions and keeping the diversity in the population. It should be noted that to preserve the best solution found, the best individual is exempt from being evaluated in the first case of the tournament.

For boundary constraint handling, the following random combination is used:

$$u_{i,j,G} = \begin{cases} (1-r)\cdot U_j + r\cdot x_{i,j,G} & \text{if } u_{i,j,G} > U_j \\ (1-r)\cdot L_j + r\cdot x_{i,j,G} & \text{if } u_{i,j,G} < L_j \end{cases} \tag{11}$$

where L_j and U_j are the lower and the upper limits of the variable x_j respectively and r is a random number $r \sim U(0, 0.1)$.

C. Parameter Adaptation

Both current-to-$p$best and Lévy flight mutations have independent parameter adaptation. However, it is necessary to extend the weighting of the parameters as to include the total amount of constraint violation of either individual for constrained optimization problems. The amount of improvement Δf_i of an individual $\boldsymbol{u}_i$ over an individual $\boldsymbol{x}_i$ is defined as follows:

$$\Delta f_i = \begin{cases} svc(\boldsymbol{x}_i;\boldsymbol{\epsilon}) - svc(\boldsymbol{u}_i;\boldsymbol{\epsilon}) & \text{If improvement is based on feasibility.} \\ f(\boldsymbol{x}_i) - f(\boldsymbol{u}_i) & \text{If improvement is based on optimality.} \end{cases} \tag{12}$$

To keep both Δf_i obtained by feasibility improvement and optimality improvement in the same range, they are divided by the maximum value of Δf_i obtained in each case. The weighted Lehmer mean [14] is computed over a set of parameter S where Δf_i is used in order to influence the adaptation (S refers to either S_{CR} or S_F for current-to-$p$best mutation or either S_{CR_L}, S_{F_L} for Lévy flight mutation).

$$L_2(S;w) = \frac{\sum_{s\in S} w_s \cdot s^2}{\sum_{s\in S} w_s \cdot s} \tag{13}$$

$$w_s = \frac{\Delta f_s}{\sum_j \Delta f_j} \tag{14}$$

Finally, the update of the cyclical memories M, in particular for the cell k, is given by

$$M_{k,G+1} = \begin{cases} L_2(S;w) & \text{if } |S| > 0. \\ M_{k,G+1} & \text{other case.} \end{cases} \tag{15}$$

M refers to either M_{CR} or M_F for current-to-$p$best mutation or either M_{CR_L}, M_{F_L} for Lévy flight mutation. Each mutation has its own index k which controls that the update is performed cyclically independently for each mutation.

D. Mutation Strategy Selection

The probability to choose a mutation operator is given by q_G. Throughout the evolutionary process, the probability q_G is adapted according to the amount of improvement that is obtained by each mutation. Considering q_G as the probability to generate new individuals using the Lévy flight mutation, its adaptation process is given by

$$q_{G+1} = \begin{cases} q_G & \text{if } |\Delta f_L| = |\Delta f| = 0. \\ \mu \cdot q_G + (1-\mu)\cdot \hat{q}_G & \text{other case.} \end{cases} \tag{16}$$

$$\hat{q}_G = \frac{\sum_{i=1}^{|\Delta f_L|} \Delta f_{Li}}{\sum_{i=1}^{|\Delta f_L|} \Delta f_{Li} + \sum_{j=1}^{|\Delta f|} \Delta f} \tag{17}$$

where $\mu \in (0,1)$ is a parameter for consider past probabilities and smooth the probability update. To avoid the degenerate case where some mutation strategy become unused during the evolutionary process, q_G is clipped in the range $(q_{min}, 1-q_{min})$ where $q_{min} \in (0, 0.5)$.

IV. Parameter Setting and Experiments

The proposed COLSHADE algorithm is tested on 57 CEC2020 Test-suite of Non-Convex Constrained Optimization Problems from the Real-World [15]. The experiments are carried out according the guidelines given in [16]: 25 independent runs are performed for each problem and a maximum number of function evaluation is given by

$$MAX_{FEs} = \begin{cases} 1\times 10^5 & D \le 10 \\ 2\times 10^5 & 10 < D \le 30 \\ 4\times 10^5 & 30 < D \le 50 \\ 8\times 10^5 & 50 < D \le 150 \\ 10^6 & \text{Other case.} \end{cases} \tag{18}$$

The COLSHADE parameter setting is summarized next:

- Population parameters: $r^{N^{init}} = 18$, $r^{arc} = 2.6$.
- Proportion of best solutions: $p = 0.11$.
- Proportion of ϵ-feasible solutions: $p_G = 0.2 \times P_G$
- Size of cyclical memories: $H = 6$.
- Stable distribution parameters: $\alpha = 0.5$, $\beta = 1$, $\gamma = 0.01$, $\delta = 0$.
- Minimum probability for mutations: $q_{min} = 10^{-3}$.
- Update probability rate: $\mu = 0.25$.
- Function evaluations with tolerance $\boldsymbol{\epsilon}_G$: $FEs_\epsilon = 0.6 \times MAX_{FEs}$.
- Final tolerance: $\epsilon_f = 10^{-4}$.

The characteristics of the computer are shown in Table I

TABLE I
PC Configuration

OS	Linux Mint 19.3
CPU	Intel(R) i7-8700 CPU @ 3.2 GHz
RAM	32 GB
Language	MATLAB (R2019a)
Algorithm	COLSHADE

V. RESULTS

The mean value of constraint violation $v(\boldsymbol{x})$ is defined as

$$v(\boldsymbol{x}) = \frac{\sum_{i=1}^{p} G_i(\boldsymbol{x}) + \sum_{j=p+1}^{m} H_j(\boldsymbol{x})}{m} \quad (19)$$

where

$$G_i(\boldsymbol{x}) = \begin{cases} g_i(\boldsymbol{x}) & \text{if } g_i(\boldsymbol{x}) > 0 \\ 0 & \text{if } g_i(\boldsymbol{x}) \leq 0 \end{cases} \quad (20)$$

$$H_j(\boldsymbol{x}) = \begin{cases} h_j(\boldsymbol{x}) & \text{if } |h_j(\boldsymbol{x})| - 10^{-4} > 0 \\ 0 & \text{if } |h_j(\boldsymbol{x})| - 10^{-4} \leq 0 \end{cases} \quad (21)$$

TABLE II
ALGORITHM COMPLEXITY OF COLSHADE

$T1(s)$	$T2(s)$	$(T2 - T1)/T1$
6.5417	13.2325	1.0228

Table II reports the algorithm complexity as requested for the competition. The experimental results for the COLSHADE Algorithm are listed in Tables VI - XI, where the values of the objective function $f(\boldsymbol{x})$ and the violation of constraints $v(\boldsymbol{x})$ for the best, median, mean and worst solutions are shown. The values of feasibility rate FR and c are defined as follows

$$FR = \frac{\text{Total feasible trials}}{\text{Total trials}} \quad (22)$$

and $c = (c_1, c_2, c_3)$ such that:

- c_1 is the number of constraints violated by an amount greater than 1.
- c_2 is the number of constraints violated by an amount in the range $[0.01, 1.0]$.
- c_3 is the number of constraints violated by an amount in the range $(0, 0.01)$.

COLSHADE found feasible solutions on 44 of 57 real-world constrained optimization problems. In particular, the most challenging real-world constrained optimization problems where no feasible solution was found are the power system problems RC34 to RC43, industrial chemical process problems R06, R07, and the livestock feed ration optimization problem RC51. COLSHADE also found 42 feasible median solutions, in addition to achieving feasibility rate $FR = 100\%$ in 39 problems.

Additionally, for the sake of better understanding our basic algorithms, the performance of COLSHADE and L-SHADE were contrasted. Since the L-SHADE is a global optimization algorithm, the constraint handling proposed in section III was adopted to handle constrained problems. The COLSHADE and L-SHADE algorithms with static constraint handling, called COLSHADE-SCH and L-SHADE-SCH respectively, were also compared. In the case of static constraint handling, $\boldsymbol{\epsilon}_{j,G} = \epsilon_f$ for all equality constraints $h_j(\boldsymbol{x})$ during all generations. The metrics used to compare the algorithms are the normalized adjusted objective function and the performance measure (PM) used for the CEC2020 [16] and the convergence speed.

The Wilcoxon's test results of the normalized adjusted objective function is summarized in Table V. The values shown in the columns R^+ and R^- represent the signed-rank sum of COLSHADE and the compared algorithm respectively and numbers in parentheses indicate the number of problems in which the result obtained by the algorithms is different. For comparing COLSHADE with algorithms with static constraint handling, only problems with at least one equality constraint are considered. The results in Table V show a significant improvement of COLSHADE over COLSHADE-SCH and L-SHADE-SCH with a level of significance $\alpha = 0.05$ in the cases of the mean and the median and over the L-SHADE with a level of significance $\alpha = 0.1$ in the case of the median.

The performance measure is presented in Table III. The results show that COLSHADE achieves the best performance measure among the four compared algorithms. Furthermore, notice that algorithms with dynamic constraint handling outperform the algorithms with static constraint handling.

The converge speed is summarized in Table IV. For convergence speed evaluation purpose, we define a *successful run* as a run in which the algorithm finds a feasible solution $\boldsymbol{x}$ that satisfies $f(\boldsymbol{x}) - f(\boldsymbol{x}^*) \leq 0.0001$. Therefore, the convergence speed is measured as the rate of function evaluations required to obtain a successful run. As shown in Table IV, COLSHADE and COLSHADE-SHC solved 31 problems. However, the first performed 646 successful runs whilst the later only 620. Therefore, the mean number of function evaluations ($\overline{\%FE}$) is a bit larger for COLSHADE.

VI. CONCLUSIONS

The COLSHADE algorithm for constrained optimization problems is introduced in this paper. It is an adaptation of the L-SHADE algorithm which has proved quite successful for global optimization. However, the basic search of our approach still is performed by a Differential Evolution. The main features of our approach are the adaptive Lévy flight and the adaptive current-to-$p$best mutation, which in turn are chosen through an adaptable probability value. The approach aims to be more robust although it can loose some convergence speed. Lévy flights are used in global optimization algorithms but we propose one of the first approaches based on adaptive Lévy flights for constrained optimization.

ACKNOWLEDGMENTS

We acknowledge support from Proyecto FORDECyT No. 296737 "Consorcio en Inteligencia Artificial" and from CONACYT (Mexico), Grant 258033.

REFERENCES

[1] X.-S. Yang and S. Deb, "Cuckoo search via lévy flights," in *2009 World congress on nature & biologically inspired computing (NaBIC)*. IEEE, 2009, pp. 210–214.

[2] X.-S. Yang, "Firefly algorithm, levy flights and global optimization," in *Research and development in intelligent systems XXVI*. Springer, 2010, pp. 209–218.

TABLE III
PERFORMANCE MEASURE

Algorithm	Performance Measure
COLSHADE	0.2777
L-SHADE	0.3086
L-SHADE-SCH	0.4363
COLSHADE-SCH	0.4372

TABLE IV
CONVERGENCE SPEED

Algorithm	$\overline{\%FEs}$	# Solved RC	# Successful runs
COLSHADE	31.50	31	646
L-SHADE	30.78	32	641
L-SHADE-SCH	30.37	32	626
COLSHADE-SCH	29.13	31	620

[3] K. Deb, “An efficient constraint handling method for genetic algorithms,” *Computer methods in applied mechanics and engineering*, vol. 186, no. 2-4, pp. 311–338, 2000.

[4] A. M. Edwards, R. A. Phillips, N. W. Watkins, M. P. Freeman, E. J. Murphy, V. Afanasyev, S. V. Buldyrev, M. G. da Luz, E. P. Raposo, H. E. Stanley *et al.*, “Revisiting lévy flight search patterns of wandering albatrosses, bumblebees and deer,” *Nature*, vol. 449, no. 7165, p. 1044, 2007.

[5] X. Yao, Y. Liu, and G. Lin, “Evolutionary programming made faster,” *IEEE Transactions on Evolutionary computation*, vol. 3, no. 2, pp. 82–102, 1999.

[6] C.-Y. Lee and X. Yao, “Evolutionary programming using mutations based on the lévy probability distribution,” *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 1, pp. 1–13, 2004.

[7] J. P. Nolan, *Stable Distributions - Models for Heavy Tailed Data*. Boston: Birkhauser, 2018, in progress, Chapter 1 online at http://fs2.american.edu/jpnolan/www/stable/stable.html.

[8] A. A. Heidari and P. Pahlavani, “An efficient modified grey wolf optimizer with lévy flight for optimization tasks,” *Applied Soft Computing*, vol. 60, pp. 115–134, 2017.

[9] 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)*. IEEE, 2014, pp. 1658–1665.

[10] 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.

[11] 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.

[12] R. Tanabe and A. Fukunaga, “Success-history based parameter adaptation for differential evolution,” in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 71–78.

[13] D. Zaharie, “Critical values for the control parameters of differential evolution algorithms,” *Proceedings of MENDEL*, vol. 2, no. December, p. 6267, 2002.

[14] F. Peng, K. Tang, G. Chen, and X. Yao, “Multi-start jade with knowledge transfer for numerical optimization,” in *2009 IEEE Congress on Evolutionary Computation*. IEEE, 2009, pp. 1889–1895.

[15] A. Kumar, G. Wu, M. Z. Ali, R. Mallipeddi, P. N. Suganthan, and S. Das, “A test-suite of non-convex constrained optimization problems from the real-world and some baseline results,” *Swarm and Evolutionary Computation*, p. 100693, 2020.

[16] P. N. Suganthan, “Guidelines for real-world single-objective constrained optimisation competition,” Mar 2020. [Online]. Available: https://github.com/P-N-Suganthan/2020-RW-Constrained-Optimisation

TABLE V
RESULTS OF WILCOXON SIGNED-RANK TEST

COLSHADE vs	Best			Mean			Median		
	R^+	R^-	$p\text{-}value$	R^+	R^-	$p\text{-}value$	R^+	R^-	$p\text{-}value$
L-SHADE	213	165	(27) 0.5640	442	338	(39) 0.4671	303	132	(29) 0.0644
COLSHADE-SCH	184	141	(25) 0.5599	373.5	91.5	(30) 0.0037	324.5	81.5	(28) 0.0056
L-SHADE-SCH	196	104	(24) 0.1887	406	90	(31) 0.0018	340	95	(29) 0.0079

TABLE VI
RESULTS FOR PROBLEMS RC01-RC09

		RC01	RC02	RC03	RC04	RC05	RC06	RC07	RC08	RC09
Best	f	189.48406	7049.037	-4529.1197	-0.388260	-400.0056	2.019732	2.043678	2	2.557655
	v	0	0	0	0	0	1.9168E-02	3.9153E-03	0	0
Median	f	210.40554	7049.037	-4529.1197	-0.388260	-391.07003	2.144757	2.183057	2	2.557655
	v	0	0	0	0	0	2.1647E-02	1.7893E-02	0	0
Mean	f	217.274350	7049.037	-4366.6773	-0.387465	-340.830432	2.063398	1.840025	2	2.557655
	v	7.0877E-06	0	0	0	0	2.1776E-02	3.9823E-02	0	0
Worst	f	209.4529	7049.037	-3716.9077	-0.373965	-5.739266	2.155065	1.704963	2	2.557655
	v	1.1277E-04	0	0	0	0	2.5211E-02	0.533290	0	0
Std	f	25.451611	0	324.8848	2.8743E-03	114.812301	0.103635	0.195037	0	0
	v	2.3334E-05	0	0	0	0	1.5769E-03	0.101060	0	0
FR(%)		88	100	100	100	100	0	0	100	100
c		(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 3 5)	(0 6 2)	(0 0 0)	(0 0 0)

TABLE VII
RESULTS FOR PROBLEMS RC10-RC17

		RC10	RC11	RC12	RC13	RC14	RC15	RC16	RC17
Best	f	1.076543	107.78178	2.924831	26887.422	58505.45	2994.4245	3.2213E-02	1.2665E-02
	v	0	0	0	0	0	0	0	0
Median	f	1.076543	168.25298	2.924831	26887.422	58505.45	2994.4245	3.2213E-02	1.2665E-02
	v	0	0.125	0	0	0	0	0	0
Mean	f	1.104296	147.815319	2.924831	26887.422	58505.45	2994.4245	3.2213E-02	1.2665E-02
	v	0	0.095	0	0	0	0	0	0
Worst	f	1.25	151.83494	2.924831	26887.422	58505.45	2994.4245	3.2213E-02	1.2666E-02
	v	0	0.125	0	0	0	0	0	0
Std	f	6.3590E-02	20.785842	4.4409E-16	3.6380E-12	7.2760E-12	4.5475E-13	0	1.0625E-07
	v	0	5.3385E-02	0	0	0	0	0	0
FR(%)		100	24	100	100	100	100	100	100
c		(0 0 0)	(0 1 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)

TABLE VIII
RESULTS FOR PROBLEMS RC18-RC25

		RC18	RC19	RC20	RC21	RC22	RC23	RC24	RC25
Best	f	6059.7143	1.6702177	263.89584	0.235242	0.525769	16.069869	2.543786	1616.1198
	v	0	0	0	0	0	0	0	0
Median	f	6059.7143	1.6702177	263.89584	0.235242	0.53	16.069869	2.543786	1616.1198
	v	0	0	0	0	0	0	0	0
Mean	f	6062.179252	1.6702177	263.89584	0.235242	0.541026	16.069869	2.543786	1639.037352
	v	0	0	0	0	0	0	0	0
Worst	f	6090.5262	1.6702177	263.89584	0.235242	0.746667	16.069869	2.543786	2129.1452
	v	0	0	0	0	0	0	0	0
Std	f	8.359059	0	0	0	4.2573E-02	0	0	100.728213
	v	0	0	0	0	0	0	0	0
FR(%)		100	100	100	100	100	100	100	100
c		(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)

TABLE IX
RESULTS FOR PROBLEMS RC26-RC33

		RC26	RC27	RC28	RC29	RC30	RC31	RC32	RC33
Best	f	35.359232	524.45076	16958.202	2964895.4	2.658559	0	-30665.539	2.639347
	v	0	0	0	0	0	0	0	0
Median	f	36.249291	524.45076	16958.202	2964895.4	2.658559	0	-30665.539	2.639347
	v	0	0	0	0	0	0	0	0
Mean	f	36.610975	524.45076	16958.202	2964895.4	2.661834	1.8807E-16	-30665.539	2.639347
	v	0	0	0	0	0	0	0	0
Worst	f	40.931153	524.45076	16958.202	2964895.4	2.699494	1.2074E-15	-30665.539	2.639347
	v	0	0	0	0	0	0	0	0
Std	f	1.367709	0	0	0	1.1105E-02	3.8137E-16	0	0
	v	0	0	0	0	0	0	0	0
FR(%)		100	100	100	100	100	100	100	100
c		(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)

TABLE X
RESULTS FOR PROBLEMS RC34-RC41

		RC34	RC35	RC36	RC37	RC38	RC39	RC40	RC41
Best	f	4.316049	101.37357	116.30845	2.712317	10.080512	11.564254	40.67832	2.187563
	v	1.2012E-03	3.8207E-02	3.6130E-02	8.9353E-03	7.8760E-03	6.5157E-03	0.531612	0.300637
Median	f	3.190423	111.05303	88.632283	2.535576	8.874342	8.216608	37.407837	45.356177
	v	4.4798E-03	0.103417	0.134630	1.5125E-02	1.3914E-02	1.4770E-02	0.869289	0.616779
Mean	f	4.954820	96.074006	84.323848	2.695820	8.277646	9.309363	111.959857	18.276486
	v	6.0654E-03	0.136460	0.156199	1.8351E-02	1.6262E-02	1.6601E-02	0.919681	0.638547
Worst	f	7.447166	98.87086	84.074297	4.315512	9.0577711	5.8016909	150.71173	52.907246
	v	1.8770E-02	0.466006	0.383872	3.6009E-02	3.7174E-02	3.1957E-02	1.590694	1.168034
Std	f	2.015848	21.297331	19.426620	0.791943	1.628060	2.539734	79.856820	14.951658
	v	4.8185E-03	9.3360E-02	9.8732E-02	8.7947E-03	7.3959E-03	6.7375E-03	0.245228	0.182121
FR(%)		0	0	0	0	0	0	0	0
c		(0 14 55)	(2 96 46)	(4 87 51)	(0 40 11)	(0 35 21)	(0 42 32)	(11 26 32)	(13 21 33)

TABLE XI
RESULTS FOR PROBLEMS RC42-RC49

		RC42	RC43	RC44	RC45	RC46	RC47	RC48	RC49
Best	f	-1.662617	15.513014	-6197.2807	3.4368E-02	2.0240E-02	1.2783E-02	1.6827E-02	2.1718E-02
	v	0.701991	0.705906	0	0	0	0	0	0
Median	f	-1.608894	19.797358	-5975.6495	4.1372E-02	2.4814E-02	1.8932E-02	2.1057E-02	3.2378E-02
	v	1.025849	1.073704	0	0	0	0	0	0
Mean	f	-2.613798	24.029476	-6032.41908	4.2795E-02	2.6082E-02	1.8212E-02	2.1876E-02	3.2582E-02
	v	1.028898	1.035738	0	0	0	0	0	0
Worst	f	-1.0213907	19.987572	-5889.0798	5.4915E-02	4.0494E-02	2.5911E-02	3.1408E-02	4.0048E-02
	v	1.438733	1.338680	0	0	0	0	0	0
Std	f	2.217248	5.485253	106.252432	5.5182E-03	5.6786E-03	3.1952E-03	3.9959E-03	4.0738E-03
	v	0.208864	0.145028	0	0	0	0	0	0
FR(%)		0	0	100	100	100	100	100	100
c		(11 30 28)	(14 29 20)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)

TABLE XII
RESULTS FOR PROBLEMS RC50-RC57

		RC50	RC51	RC52	RC53	RC54	RC55	RC56	RC57
Best	f	2.0450E-02	4550.914	3352.2532	5029.5203	4241.665	6698.0095	14753.701	3295.0806
	v	0	2.8225E-06	0	0	0	0	0	0
Median	f	3.9494E-02	4550.9086	3370.1332	5093.8367	4245.0499	6714.5247	14759.4	3508.529
	v	0	2.8225E-06	0	0	0	0	7.0711E-05	0
Mean	f	6.5091E-02	4550.945116	3372.124748	5109.499652	4245.936432	6732.505332	14646.65576	3628.2399
	v	4.02E-05	2.8229E-06	0	0	0	1.8116E-05	1.5186E-04	0
Worst	f	0.109993	4550.9041	3400.9498	5241.4407	4254.6273	6922.468	1.4013E+04	4527.8684
	v	1.0040E-03	2.8329E-06	0	0	0	1.6763E-04	6.2475E-04	0
Std	f	4.8199E-02	6.7845E-02	12.979437	56.510696	3.409109	54.665073	204.732456	293.211450
	v	1.9675E-04	2.0498E-09	0	0	0	4.7103E-05	1.8970E-04	0
FR(%)		96	0	100	100	100	84	48	100
c		(0 0 0)	(0 0 1)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 2)	(0 0 0)