Success Rate-based Adaptive Differential Evolution L-SRTDE for GNBG 2024 Competition

1st Vladimir Stanovov
Institute of Informatics and Telecommunication
Reshetnev Siberian State University
of Science and Technology
Krasnoyarsk, Russian Federation
vladimirstanovov@yandex.ru

***Abstract*—This document describes the L-SRTDE algorithm, originally developed for the CEC 2024 competition, and applied to GNBG 2024 competition [1].**

***Index Terms*—differential evolution, numerical optimization, parameter adaptation**

I. DE MODIFICATIONS

In [2] the rank-based selection was proposed, and in [3] it was used in the L-SHADE-RSP algorithm, in which the indexes for the current-to-pbest mutation strategy were chosen with probabilities, dependent on the fitness values of corresponding individuals. For this purpose the individuals were sorted and ranked, with ranks assigned as follows:

$$rank_i = e^{\frac{-kp \cdot i}{N}}, \tag{1}$$

where kp is a selective pressure parameter, $i = 1, 2, ...N$. The selective pressure was shown to improve the convergence speed, especially in high-dimensional cases.

In [4], differential evolution with unlimited population size was studied, where all previous solutions were saved. It was shown that applying different selection strategies may result in significant performance benefits. Inspired by this study, in [5] the L-NTADE algorithm was proposed.

The L-NTADE algorithm has two populations: one for newest solutions x^{new} and another for top solutions x^{top}, having the same size N. At initialization both populations are filled with the same solutions. The mutation strategy is called r-new-to-ptop/n/t, and works as follows:

$$v_{i,j} = x^{new}_{r1,j} + F \cdot (x^{top}_{pbest,j} - x^{new}_{i,j}) + F \cdot (x^{new}_{r2,j} - x^{top}_{r3,j}) \tag{2}$$

where $r1$ and $r3$ are chosen randomly, and $r2$ is chosen with rank-based selective pressure with $kp = 3$, $pbest$ is chosen from $pb\%$ best solutions in the top population. Unlike current-to-pbest, this strategy uses a randomly chosen solution x^{new}_{r1} as a basic one, instead of target vector.

The crossover and bound constraint handling are classical in L-NTADE, but the selection step differs significantly and works as follows:

$$x_{nc} = \begin{cases} u_i, & \text{if } f(u_i) \leq f(x^{new}_{r1}) \\ x_{nc}, & \text{if } f(u_i) > f(x^{new}_{r1}) \end{cases}. \tag{3}$$

where nc is an index, iterated after every successful replacement, and if $nc > N$ it is reset to 1. The trial vector u_i is compared to the basic vector x^{new}_{r1}, and the trial is better, than it replaces individual with index nc, not with index i. This results in a continuous update of the x^{new} population with latest solutions, although better vectors may be replaced by worse ones. In addition, all successful vectors are stored in a temporary pool x^{temp} along with their fitness values. At the end of the generation the population of best solutions x^{top} is joined with x^{temp}, sorted by fitness, and the best N solutions (N is recalculated according to LPSR) are placed to x^{top}. Thus, the top population always contains N best solutions so far.

The L-NTADE algorithm was shown to perform significantly different from the classical DE scheme, and much better in multiple scenarios. However, it still relied on the success-history adaptation for setting F and Cr. In [6] the genetic programming algorithm was applied to automatically generate parameter adaptation heuristics for L-NTADE, following the Hyper-Heuristic (HH) paradigm [7]. It was found that the ratio of improved solutions in selection (i.e. success rate) has a strong connection to the setting of the scaling factor F. The current study builds on these findings, as well as experiments with curves search in [8], and proposes the L-SRTDE algorithm, described in the next section.

II. PROPOSED APPROACH: L-SRTDE

The proposed approach, described below, is based on the experiments with hyper-heuristic search for parameter adaptation methods, performed in [9]. The L-SRTDE algorithm is a modification of the L-NTADE with two main features. First, the scaling factor adaptation is based on the success rate, with the mean for sampling set as follows:

$$mF = 0.4 + 0.25 \cdot tanh(5 \cdot SR) \tag{4}$$

where $SR = \frac{NS}{N}$, NS is the number of successful solutions, i.e. number of times the selection worked. The mF value is then used to sample F values for each mutation as $F = randn(mF, 0.02)$. The described dependence was found by a hyper-heuristic approach, similar to the one used in [8].

The second modification uses success rate to set the pb parameter for the mutation strategy. The dependence is as follows:

$$pb = 0.7 \cdot e^{-7 \cdot SR} \tag{5}$$

That is, if the success rate is close to 0, then the $pbest$ index is chosen from 70% of the best solutions, which is close to random choice. However, as the success rate grows, the pb decreases, and focuses more on smaller number of best solutions. Note that even if $pb \cdot N < 2$, the $pbest$ index is chosen from one of two best solutions, i.e. $pbest = randi(max(2, pb \cdot N))$, where $randi(m)$ is a random integer from 1 to m.

Another minor modification in L-SRTDE is the usage of repaired crossover rate, described in [10]. Instead of using the crossover rate Cr, sampled for crossover, the actual Cr_a value is calculated, i.e. the number of replaced components, divided by dimensionality D. In this case during adaptation $M_{Cr,h}$ can never go to 0, and will stay at least at $1/D$. For Cr adaptation the SHA is used in L-SRTDE, and the values are sampled with smaller dispersion: $Cr = randn(M_{Cr,k}, 0.05)$.

Figure 1 shows the curves that describe the dependence of mF and pb values on success rate.

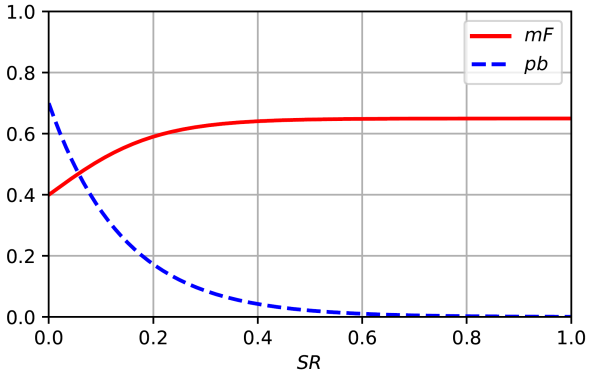

Fig. 1. Dependence of mF and pb values on success rate.

The full pseudocode of L-SRTDE is given in Algorithm 1.

The next section contrains the experimental setup and numerical results.

III. Experiments and Results

The GNBG 2024 competition consists of 24 test functions with 30 variables each, and the computational resource is set to 500,000 for f_1 to f_{15} and 1,000,000 for f_{16} to f_{24}.

The parameters of the L-SRTDE were set as follows: initial population size $N = 10D$, minimal population size $N_{min} = 4$, number of memory cells $H = 5$, initial values in memory cells $M_{Cr,k} = 1$, selective pressure parameter $kp = 3$. These values did not change between the benchmarks. The L-SRTDE algorithm was implemented in C++, compiled with GCC and ran under Ubuntu 20.04 machine with AMD Ryzen 5800X processor. The results post-processing, ranking and statistical tests were performed with Python 3.8.

Algorithm 1 L-SRTDE

Input: D, NFE_{max}, N_{max}, target function $f(x)$
Output: x^{top}_{best}, $f(x^{top}_{best})$
Set $N = N_{max}$, $N_{min} = 4$, $H = 5$, $M_{Cr,r} = 1$
Set $SR = 0.5$, $k = 1$, $g = 0$, $nc = 1$, $kp = 3$
Initialize population $(x^{new}_{1,j}, ..., x^{new}_{N,j})$ randomly
Calculate $f(x^{new})$
Copy x^{new} to x^{top}, $f(x^{new})$ to $f(x^{top})$
while $NFE < NFE_{max}$ **do**
  $S_{Cr} = \emptyset$, $S_{\Delta f} = \emptyset$
  Sort x^{new} and $f(x^{new})$
  Assign ranks for x^{new}
  for $i = 1$ to N^g **do**
    $mF = 0.4 + 0.25 \cdot th(5 \cdot SR)$
    repeat
      $F_i = randn(mF, 0.02)$
    until $F_i \in (0, 1)$
    Current memory index $r = randi[1, H]$
    Crossover rate $Cr_i = randn(M_{Cr,r}, 0.05)$
    $Cr_i = min(1, max(0, Cr))$
    $r1 = randi(N^g)$
    $pb = 0.7 \cdot e^{-7 \cdot SR}$
    repeat
      $pbest = randi(1, N^g \cdot pb)$
      Generate $r2$ with rank-based selection
      $r3 = randi(1, N^g)$
    until indexes $r1$, $r2$, $r3$ and $pbest$ are different
    Apply mutation to produce v_i with F_i
    Apply binomial crossover to produce u_i with Cr_i
    Calculate actual rate Cr_a
    Apply bound constraint handling method
    Calculate $f(u_i)$
    if $f(u_i) < f(x^{new}_{r1})$ **then**
      $Cr_a \rightarrow S_{Cr}$
      $(f(x^{new}_{r1}) - f(u_i)) \rightarrow S_{\Delta f}$
      $u_i \rightarrow x^{temp}$
      $f(x^{temp}_m) = f(u_i)$
      $x^{new}_{nc} = u_i$
      $f(x^{new}_{nc}) = f(u_i)$
      $nc = mod(nc + 1, N^g)$, $m = m + 1$
    end if
  end for
  Get N^{g+1} with LPSR
  Join together x^{top} and x^{temp} and sort
  Copy N^{g+1} best vectors back to x^{top}
  if $N^g > N^{g+1}$ **then**
    Remove worst individuals from x^{new}
  end if
  Update $M_{Cr,k}$
  $k = mod(k + 1, H)$, $g = g + 1$, $m = 1$
end while
Return x^{top}_{best}, $f(x^{top}_{best})$

A. GNBG 2024 results

Table I containts the experimental results.

Figures 2 shows the convergence graphs of the considered algorithm.

IV. Conclusion

As the comparison table shows, the L-SRTDE algorithm was able to solve 10 problems with 100% success rate out of 24 problems.

References

[1] A. H. Gandomi, K. Deb, D. Yazdani, R. Salgotra, and M. N. Omidvar, "Gecco 2024 numerical global optimization competition on gnbg-generated test suite," 2024. [Online]. Available: https://competition-hub.github.io/GNBG-Competition/

[2] W. Gong and Z. Cai, "Differential evolution with ranking-based mutation operators," *IEEE Transactions on Cybernetics*, vol. 43, pp. 2066–2081, 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:1755811

[3] V. Stanovov, S. Akhmedova, and E. Semenkin, "Lshade algorithm with rank-based selective pressure strategy for solving cec 2017 benchmark problems," *2018 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, 2018.

[4] T. Kitamura and A. Fukunaga, "Differential evolution with an unbounded population," *2022 IEEE Congress on Evolutionary Computation (CEC)*, 2022.

[5] V. Stanovov, S. Akhmedova, and E. Semenkin, "Dual-population adaptive differential evolution algorithm l-ntade," *Mathematics*, 2022.

[6] V. Stanovov and E. Semenkin, "Genetic programming for automatic design of parameter adaptation in dual-population differential evolution," *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:260119279

[7] E. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. Woodward, *A Classification of Hyper-Heuristic Approaches: Revisited*. Springer International Publishing, 2019, pp. 453–477.

[8] V. Stanovov and E. Semenkin, "Surrogate-assisted automatic parameter adaptation design for differential evolution," *Mathematics*, vol. 11, no. 13, 2023. [Online]. Available: https://www.mdpi.com/2227-7390/11/13/2937

[9] V. Stanovov, L. Kazakovtsev, and E. Semenkin, "Hyper-heuristic approach for tuning parameter adaptation in differential evolution," *Axioms*, vol. 13, p. 59, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:267108274

[10] W. Gong, Z. Cai, and Y. Wang, "Repairing the crossover rate in adaptive differential evolution," *Appl. Soft Comput.*, vol. 15, pp. 149–168, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:41345493

TABLE I
RESULTS OF L-SRTDE ON GNBG 2024

Func	Absolute error	Required FEs to Acceptance Threshold	Success rate
F1	0 ± 0	67714.2 ± 975.567	1
F2	0 ± 0	182631 ± 2207.67	1
F3	0 ± 0	94520.7 ± 1351.24	1
F4	0 ± 0	73816.1 ± 841.001	1
F5	0.0154979 ± 0.0287145	277624 ± 120136	0.774194
F6	0.064579 ± 0.0384646	449735 ± 93110.7	0.225806
F7	0 ± 0	88267.5 ± 1135.51	1
F8	0 ± 0	88701.8 ± 1486.91	1
F9	$1.70645e-09 \pm 7.21621e-09$	293285 ± 6441.9	0.935484
F10	0 ± 0	101405 ± 2311.76	1
F11	0 ± 0	100190 ± 2196.15	1
F12	0 ± 0	100567 ± 1368.95	1
F13	$4.08149e-07 \pm 9.95954e-07$	386527 ± 100803	0.419355
F14	1.19837 ± 1.74367	500000 ± 0	0
F15	3.03681 ± 0.215713	500000 ± 0	0
F16	632.113 ± 125.876	970092 ± 163813	0.0322581
F17	619.358 ± 43.611	$1e+06 \pm 0$	0
F18	586.835 ± 165.996	942508 ± 218923	0.0645161
F19	565.042 ± 192.984	916868 ± 253973	0.0967742
F20	$6.90758e-09 \pm 2.12677e-08$	293478 ± 4444.26	0.903226
F21	5 ± 0	$1e+06 \pm 0$	0
F22	50 ± 0	$1e+06 \pm 0$	0
F23	0 ± 0	193779 ± 2910.79	1
F24	1.10427 ± 0.406818	938082 ± 189174	0

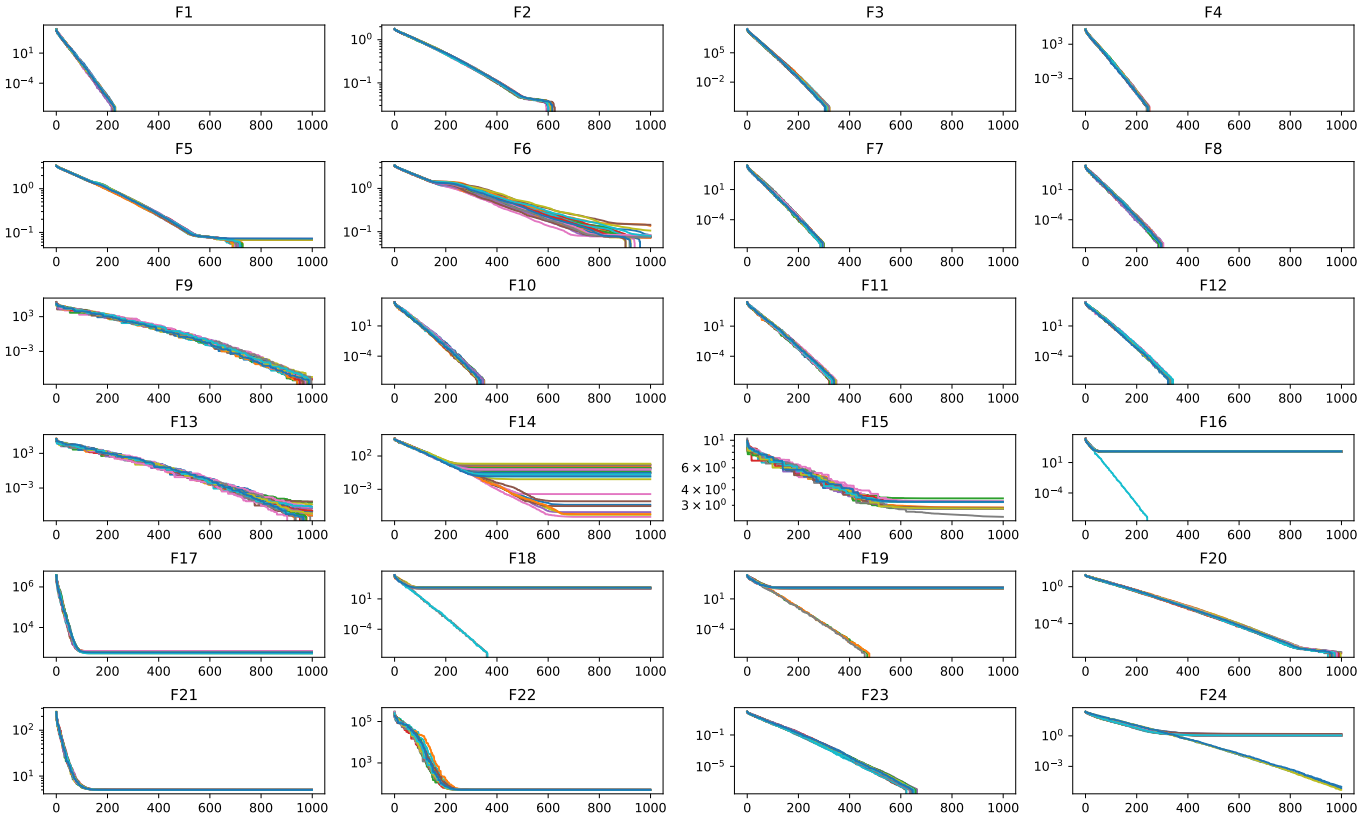

Fig. 2. Convergence graphs of L-SRTDE tested on GNBG 2024