An Enhanced EDBF Framework: Constraint-Law-Method (CLM) for Improving Multi-parent Crossover Algorithms

Zhengkang ZUO (1801110646@pku.edu.cn)
Peking University https://orcid.org/0000-0003-1255-6332

Research Article

Keywords: CLM. Enhanced EDBF. Multi-parent crossover algorithms. Optimization.

Posted Date: July 6th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1795740/v1

An Enhanced EDBF Framework: Constraint-Law-Method (CLM) for Improving Multi-parent Crossover Algorithms

Zhengkang Zuo[1, *]

[1] School of Earth and Space Science, Peking University, Beijing, 100871 China.

Corresponding author: Zhengkang Zuo (e-mail: zuozhengkang@gmail.com)

Abstract: Empirical distribution-based framework (EDBF), as a general framework utilized in lots of muti-parent crossover algorithms (MCAs), makes MCAs much more efficient at each iteration. However, EDBF cannot work with the numerous parent chromosomes especially exceeding fifty. To address this problem, an enhanced EDBF framework, namely constraint-law-method (CLM), is proposed by adaptively changing the boundary of weight assigned to each parent chromosome according to the constraint law. Furthermore, CLM is compared with EDBF, ABPSO and RCBBFA algorithms on 20 benchmark functions in terms of convergence, efficiency and accuracy. Experimental results demonstrate that CLM outperforms comparative algorithms on most of benchmark functions. As a general framework rather than a specific algorithm, CLM is easy to implement and can easily be accommodated to any existing MCAs. Finally, the C++ source code is available at https://github.com/ZhengkangZUO-2020/CLM-Framework-Codes.

Keywords: CLM. Enhanced EDBF. Multi-parent crossover algorithms. Optimization.

1. Introduction

Muti-parent crossover algorithms (MCAs) are widely used in solving optimization problems in many fields relying on encoding, crossover, variation and choice operators to produce iterative offspring chromosome. At each crossover stage, a linear combination of weights at the same scale hybridizes a fixed number of parent chromosomes. If parent chromosomes are iteratively selected, the weights will indicate these papent chromosomes how to propagate. Essentially, all MCAs differentiate one algorithm from others through weight boundary. For example, each weight of the genetic algorithm (GA) [1] ranges from -0.5 to 1.5 with sum of 1. Moreover, adjusting the boundaries of weights shifts GA to unified multi-parent combination algorithm (UMCA) [2], particle swarm optimization (PSO) [3] and differential evolution (DE) [4], respectively. Specifically, each weight of UMCA varies from -1 to 1.5 and with sum shifts from -1 to 2, whereas in PSO, each weight varies from 0 to 2 and with sum shifts from 0 to 4. Besides, each weight of DE varies from 0 to 1 with sum shifts from 0 to 1.

However, existing MCAs have no capacity to efficiently spawn weights that satisfy the above boundaries, which further curbed their extensive use for general applications. To address this problem, empirical distribution-based framework (EDBF) was proposed [5, 6] and utilized in elite-subspace multi-parent crossover algorithms (EMCA) [7], global-local mixed evolutionary algorithm (GLME) [8] and GLME based on domain decomposition (GLME-DD) [9]. Besides, EDBF is also applied in certain applications, such as Epidemiology [10], Climatology [11, 12] and Photogrammetry [13]. Even if EDBF demenstrates a good performance, it cannot work with the numerous parent chromosomes especially exceeding fifty.

Therefore, this study proposed an enhanced EDBF framework, namely constraint-law-method (CLM), by adaptively changing the weight boundary of each parent chromosome according to the constraint law. Furthermore, CLM was compared with EDBF [5], ABPSO [14] and RCBBFA [15] algorithms on 20 benchmark functions [16] in terms of accuracy, convergence and efficiency. As a general framework rather than a specific algorithm, CLM is easy to implement and can easily be accommodated to any existing MCAs.

The rest of this paper is organized as follows: Section 2 describes the MCAs algorithm. Section 3 introduces the EDBF framework in MCAs. In Section 4, we proposed a CLM framework. Experimental studies are presented in Section 5. Finally, the paper is concluded in Section 6.

2. MCAs

MCAs contain four operators, such as encode, crossover, mutation and selection jointly to spawn offspring-solutions, and its flowchart is generalized in Figure 1. Consider the optimization problem:

$$Z = \underset{x \in D}{\text{Min}} f(X),$$

where the variable $X = (x_1, x_2, ..., x_n)^T \in R^n$, $D = \{X | I_i \le x_i \le U_i, i = 1, 2, ..., n\}$ satisfies the inequality $g_i(X) \le 0, i = 1, 2, ..., q$, and $f(X)$ is the objective function. Let the m points in D be $X_j^{'}, j = 1, 2, ..., m$, note that the subspace they span is:

$$V = \left\{ X \in D \middle| X = \sum_{i=1}^{m} \alpha_i X_i^{'} \right\},$$

where the weight α_i satisfies the condition $\sum_{i=1}^{m} \alpha_1 = 1, -0.5 \le \alpha_i \le 1.5$.

Note:

$$h_i(X) = \begin{cases} 0, g_i(X) \le 0 \\ g_i(X), \text{otherwise} \end{cases},$$

$$H(X) = \sum_{i=1}^{q} h_i(X)$$

Define the logistic funtion:

$$\text{better}(X_1, X_2) = \begin{cases} H(X_1) \le H(X_2), \text{True} \\ H(X_1) > H(X_2), \text{False} \\ (H(X_1) = H(X_2)) \wedge (f(X_1) \le f(X_2)), \text{True} \\ (H(X_1) = H(X_2)) \wedge (f(X_1) > f(X_2)), \text{False} \end{cases},$$

demenstrates that X_1 outperforms X_2.

The program of MCAs is described as follows:

First step: Randomly generate an initial population $P(0) = \{X_1, X_2, ..., X_N\}$ in the search space S (they are usually required to be evenly distributed in S), $t = 0$.

Second step: Sort the population $P(t)$ according to the logistic function $\text{better}(X_1, X_2)$ from good to bad. After sorting, it is still noted as $P(t) = \{X_1, X_2, ..., X_N\}$, X_1 and X_N are the best and the worst individual, respectively.

Third step: Go to step 5 if $\text{better}(X_{\text{worst}}, X_{\text{best}})$.

Forth step: Select $K(K \le M)$ best individuals $X_1^{'}, X_2^{'}, ..., X_K^{'}$ from $P(t)$, where M represents the number of parent chromosomes involved in the hybridization. $M - K$ individuals $X_{K+1}^{'}, X_{K+2}^{'}, ..., X_M^{'}$ are randomly selected from the remaining $N - K$ individuals. These $M(M \le N)$ individuals form a subspace

$V=\left\{X \middle| X \in S, X=\sum_{i=1}^{M} \alpha_i X_i^{'}\right\}$, where the weight α_i satisfies the condition $\sum_{i=1}^{M} \alpha_i = 1, -0.5 \le \alpha_i \le 1.5$.

Randomly select L points in the subspace V to obtain L new individuals, and select the best individual $\overline{X}$ among L individuals. If $\text{better}\left(\overline{X}, X_{\text{worst}}\right)$, replace X_{worst} with $\overline{X}$ and form a new population $P(t+1)$, otherwise $P(t+1)=P(t)$. Let $t=t+1$ and go to step 2.

Fifth step: Output the best solution X_{best} and end the program of MCAs.

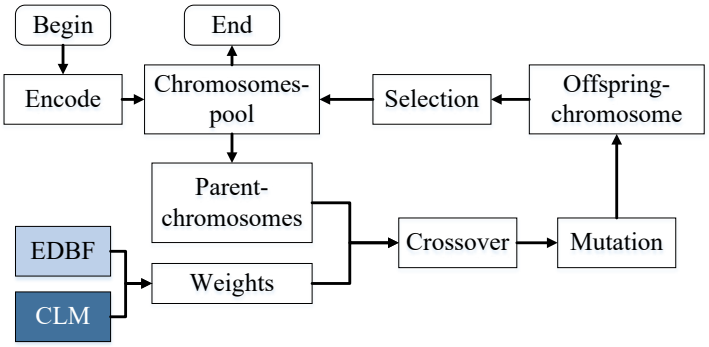

Figure 1. The flowchart of MCAs.

3. EDBF Framework in MCAs

EDBF [5] is a general framework rather than a specific algorithm, which is easy to implement and accommodate any existing MCAs, and it plays a vital role at each crossover step to accelerate the production of weights that are used for combination with parent chromosomes. The pseudocode of EDBF framework is shown in Table 1, where $P(\cdot)$ is the empirical probability of the weight spawning in the range of $[\cdot]$. In EDBF, all weights spawn into three ranges ([-0.5, 0], [0, 1], [1, 1.5]) with the empirical probability. On the contrary, existing MCAs spawn weights following a uniform distribution ($P(\alpha \in [-0.5,0])$=25%, $P(\alpha \in [0,1])$=50%, $P(\alpha \in [-0.5,1])$=75%), and the challenge is that the number of parent chromosomes (note as M) cannot exceed 13; otherwise, the program will collapse with a slight increase of M. Figure 2 shows that EDBF framework improved MCAs a lot in efficiency at each iteration, especially with much more parent chromosomes joining the optimization process.

Table 1. The pseudocode of EDBF framework

Input: $P(\alpha \in [-0.5,0]) = -0.8586 \times \mathrm{M}^{-0.9424} + 0.6115$, $P(\alpha \in [0,1]) = 0.5802 \times \mathrm{M}^{-0.8598} + 0.3442$, $P(\alpha \in [-0.5,1]) = P(\alpha \in [-0.5,0]) + P(\alpha \in [0,1])$.
Output: The qualified weight vector $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_M\}$
Algorithm: EDBF
stop:=($\alpha_M \in [-0.5,1.5]$)
while (not stop)
    s ← 0
    i ← 1
    stop: = (i > M − 1)
    while (not stop)
        randomly generate r in the range of [0,1]
        randomly generate α_i in the range of [-0.5,0], [0,1] and [1,1.5] in the case of $r \le P(\alpha \in [-0.5,0])$, $P(\alpha \in [-0.5,0]) < r \le P(\alpha \in [-0.5,1])$ and $r > P(\alpha \in [-0.5,1])$, respectively
        let s←s+α_i
    end while
    let $\alpha_M = 1 - s$
end while

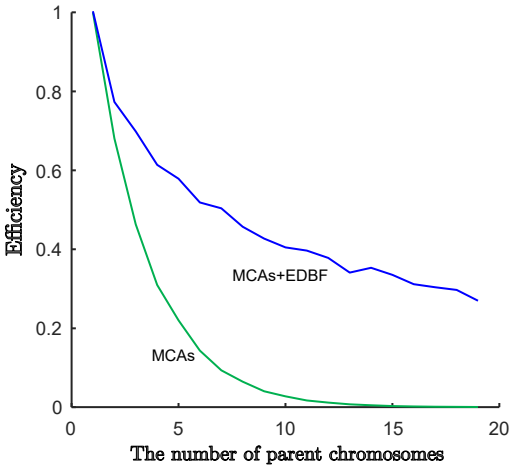

Figure 2. Efficiency comparison in weights production before and after using EDBF framework in MCAs.

4. CLM Framework in MCAs

The constraint law method (CLM) is an enhanced EDBF framework for improving MCAs by adaptively changing the boundary of weight assigned to each parent chromosome according to the constraint law, and its pseudocode is shown in Table 2. Compared with EDBF that spawn weights in different ranges bounded by the empirical probability of $P(\cdot)$, CLM spawn weights in the adaptive range of $[t_{low}, t_{up}]$. After the implementation of CLM, just once loop could spawn the weights that satisfy the boundrary, as shown in Figure 3. It could be observed that CLM outperforms EDBF a lot. For example, MCAs collapses when the number of parent chromosomes exceeds 13, and EDBF could delay the collapse time but also collapses with the number of parent chromosomes in excess of 50, but CLM avoids the collapse even with numerous parent chromosomes. In this regard, CLM supports the infinite recombination of the parent chromosomes to inherit more excellent genes to the offspring chromosome. In addition, more parent chromosomes are propitious to form the more satisfied search space to find the global optima.

Table 2. The pseudocode of CLM framework

Input: A randomly generated weight α_1 in the range of [-0.5,1.5]
Output: The qualified weight vector $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_M\}$
Algorithm: CLM
$s \leftarrow 0$
$i \leftarrow 2$
stop: = $(i > M - 1)$
while (not stop)
    $s \mathrel{+}= \alpha_{i-1}$
    $t_{low} \leftarrow -0.5 - s$
    $t_{up} \leftarrow 1.5 - s$
    $t_{low} \leftarrow \max\{t_{low}, -0.5\}$
    $t_{up} \leftarrow \min\{t_{up}, 1.5\}$
    randomly generate α_i in the range $[t_{low}, t_{up}]$
end while
let $S \leftarrow s + \alpha_{M-1}$
$\alpha_M = 1 - S$

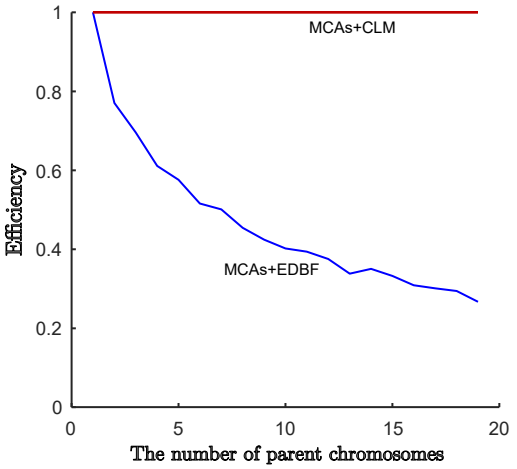

Figure 3. Efficiency comparison in weights production after using EDBF and CLM framework in MCAs.

5. Numerical Experiments

In this section, CLM framework was compared with EMCA, EDBF, ABPSO and RCBBFA algorithms on 20 benchmark functions in terms of accuracy, efficiency and convergence. To make the comparison fairer, all experimental results are averaged after 100 independent runs. Besides, all comparative algorithms have the same termination condition, that is the difference of the best-solution and the worst-solution not in excess of 1E-14. To guarantee that all comparative algorithms achieve satisfactory performance for most benchmark functions, we take the suggestions of their corresponding literature to set the values of the related parameters, except for the termination condition. Finally, the parameters of CLM are set as 100 pool chromosomes, 15 parent chromosomes, and 5 elitist chromosomes. Moreover, all experimental results are available at https://github.com/ZhengkangZUO-2020/Experiment-Results-of-CLM-Framework-.

5.1. Benchmark Function

Benchmark functions are shown in Table 3, where f_1 to f_5 are unimodal, f_6 is the discontinuous step function with one minimum, f_7 is a noisy quartic function in which *random* [0, 1) is a uniformly distributed random variable in [0, 1). Function f_8 to f_{13} are multimodal functions where the number of local minima increases exponentially with the problem dimension. Function f_{14} to f_{20} are low-dimensional functions which have only a few local minima. More details about this suite of benchmark functions could be found in references [16].

5.2. PC Configuration

The configure of all experiments is the operation system (OS) of Windows 64 bit, the RAM of 15.91 GB, and the CPU of Inter (R) Core (TM) i7-4790 CPU @ 3.60 GHz. Onward, C++ implements all versions of algorithm with the Microsoft Visual Studio 2015 complier.

5.3. Results and Discussion

5.3.1. Accuracy Comparison

In Table 4, it could be observed on experimental results of f_1—f_4, f_7, f_{10}, f_{12} and f_{13} that CLM outperforms EDBF, EMCA and RCBBFA, but slightly inferior to ABPSO. Nevertheless, the accuracies of CLM are both in excess of 1E-15, which is quite close to the global optima '0'. Besides, CLM converges to 2.84E-15 when solving f_5, which outperforms the other four comparative algorithms. Moreover, CLM exhibits the same accuracy as EMCA and ABPSO on f_{11}, which both outperform EDBF and RCBBFA. Also, CLM shows the same accuracy as EMCA, EDBF and ABPSO on f_9 and f_{14}—f_{20}, which both outperform RCBBFA. When solving f_6, CLM exhibits the same accuracy as the other four comparative algorithms, all converging to the global optima '0'. It is worthy to mention that CLM outperforms EDBF or exhibits the same accuracy as EDBF on all benchmark functions.

Table 3. Benchmark functions

Expression	n	S	Optimum
$f_1(x) = \sum_{i=1}^{n} x_i^2$	10	-100, 100	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	10	-10, 10	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	10	-100, 100	0
$f_4(x) = \max_i \{ \lvert x_i \rvert, 1 \le i \le n \}$	30	-100, 100	0
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	10	-30, 30	0
$f_6(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	10	-100, 100	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	10	-1.28, 1.28	0
$f_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	2	-500, 500	-837.966
$f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	2	-5.12, 5.12	0
$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos 2\pi x_i\right) + 20 + e$	10	-32, 32	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	2	-600, 600	0
$f_{12}(x) = \frac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$	10	-50, 50	0
$f_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	10	-50, 50	0
$f_{14}(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right]^{-1}$	2	-65.536, 65.536	1
$f_{15}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	-5, 5	0.0003075
$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	-5, 5	-1.0316285
$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	[-5,10]×[0,15]	0.398
$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	-2, 2	3
$f_{19}(x) = \sum_{i=1}^{n} [(\ln(x_i - 2))^2 + (\ln(10 - x_i))^2] - \left(\prod_{i=1}^{n} x_i\right)^{0.2}$	10	2, 10	-45.778469
$f_{20}(x) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right)$	10	-1, 1	-1

Table 4. Average accuracy of CLM, EMCA, EDBF, RCBBFA and ABPSO for 20 benchmark functions.

Benchmark Functions	Average minimum after 100 independent runs					Minimum
	EMCA+CLM	EMCA	EMCA+EDBF	RCBBFA	ABPSO	
f_1	3.02E-15	3.1560E-15	4.29E-15	0.181517	**4.28E-26**	0.00E+00
f_2	7.11E-15	7.3430E-15	1.22E-14	1.63109	**2.83E-15**	0.00E+00
f_3	2.11E-15	6.4290E-15	4.00E-15	37.1221	**4.35E-25**	0.00E+00
f_4	8.57E-15	1.2849E-14	1.27E-14	0.363689	**1.54E-26**	0.00E+00
f_5	**2.84E-15**	6.0610E-15	3.55E-15	43.436	4.81417	0.00E+00
f_6	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.00E+00
f_7	6.09E-16	7.84E-16	1.25E-15	0.254314	**1.08E-179**	0.00E+00
f_8	**-837.966**	**-837.966**	**-837.966**	-798.438	**-837.966**	-837.966
f_9	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.050112	**0.00E+00**	0.00E+00
f_{10}	7.55E-15	7.55E-15	1.47E-14	6.01E-01	**4.00E-15**	0.00E+00
f_{11}	**0.00E+00**	**0.00E+00**	1.11E-16	0.04027	**0.00E+00**	0.00E+00
f_{12}	3.68E-15	5.841E-15	1.92E-15	1.97659	**4.71E-32**	0.00E+00
f_{13}	3.38E-15	5.093E-15	2.48E-15	0.150806	**1.35E-32**	0.00E+00
f_{14}	**0.999998**	**0.999998**	**0.999998**	1.00631	**0.999998**	1
f_{15}	**0.004361**	**0.004361**	**0.004361**	0.076985	**0.004361**	0.0003075
f_{16}	**-1.03163**	**-1.03163**	**-1.03163**	-1.03103	**-1.03163**	-1.0316285
f_{17}	**0.397887**	**0.397887**	**0.397887**	0.435426	**0.397887**	0.398
f_{18}	**3**	**3**	**3**	4.35196	**3**	3
f_{19}	**-45.7785**	**-45.7785**	**-45.7785**	-25.2299	**-45.7785**	-45.778469
f_{20}	**-1**	**-1**	**-1**	-0.93477	**-1**	-1

5.3.2. Convergence Comparison

In Figure 4, it could be found that CLM outperforms EDBF and EMCA on all benchmark functions, but inferior to ABPSO and RCBBFA. However, except for f_6, RCBBFA stunk into the local optima on all the other functions. In this regard, CLM outperforms RCBBFA in terms of convergence on the other 19 benchmark functions except for f_6. Besides, CLM outperforms all the comparative algorithms on f_5 because ABPSO also stuck into the local optimum '4.81417'. Although ABPSO outperforms CLM on most of functions in terms of convergence iterations, CLM outperforms ABPSO on all the benchmark functions in terms of runtime, which will be discussed in Section 5.3.3. When solving f_{17}, the iterations of CLM reduced by almost 50 times than EDBF. CLM converged at 3585 iterations, whereas EDBF converged at 177731 iterations. Besides, CLM converged at 5219 iterations on f_{16}, decreased by almost 12 times than EDBF (63781 iterations). On f_7, the number of iterations of CLM dropped by 5.5 times than EDBF, followed by f_1 (5.3 times), f_{12} (5.2 times), f_{13} (5.1 times), f_{10} (5.1 times) and f_2 (5 times). In addition, CLM outperforms EDBF by 4.2 times and 4.1 times on f_5 and f_6 in terms of convergence iterations, respectively. On f_{20}, CLM converged prior to EDBF by 3.4 time, followed by f_9 (2.6 times), f_{15} (2.5 times) and f_4 (2.5 times). Slight superiority of CLM also could be observed on f_8, f_{14} and f_{19}. Finally, on f_3 and f_{18}, the superiority of CLM is not obvious. As a summary, CLM outperforms EDBF on all benchmark functions in terms of convergence.

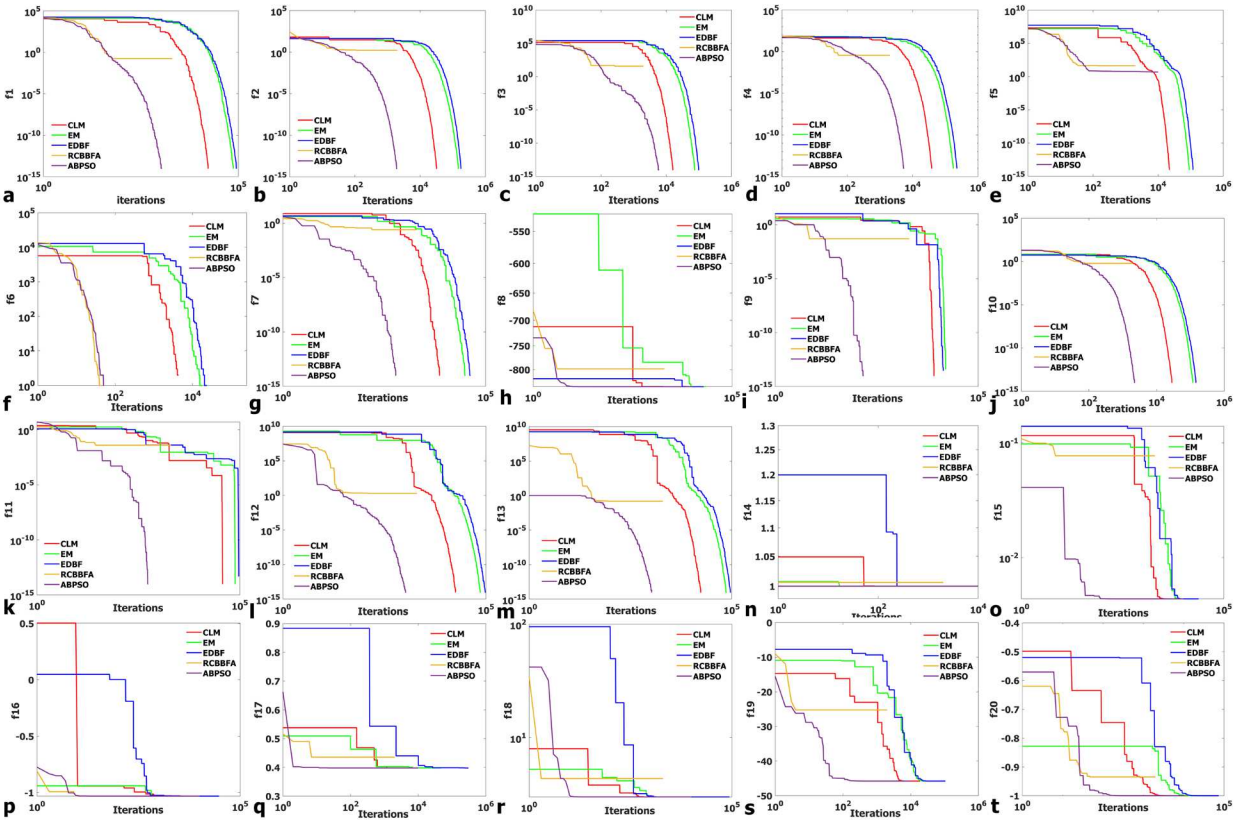

Figure 4. The comparison of convergence among EMCA, EDBF, CLM, RCBBFA and ABPSO on f_1 to f_{20}.

5.3.3. Efficiency Comparison

In Table 5, it could be observed that CLM outperforms all comparative algorithms on all benchmark functions. Specifically, CLM accelerated EMCA and EDBF the most when solving f_{17}, by 104.87 times and 69.93 times, respectively. When solving f_{14}, the efficiency of CLM outperformed RCBBFA and ABPSO by the most extent, reaching 300.47 times and 101.84 times, respectively. Compared with EMCA, CLM performed the worst on f_{14}, but also outperformed EMCA by 11.22 times. On the contrary, compared with EDBF, RCBBFA and ABPSO, CLM performed the worst on f_{18} and only outperformed EDBF by 1.65 times, RCBBFA by 2.54 times and ABPSO by 1.59 times, respectively. It is worthy to mention that CLM outperformed RCBBFA by the average of 39.05 times on all benchmark functions, followed by EMCA (37.48 times), ABPSO (23.60 times) and EDBF (8.30 times). In brief, CLM outperforms EMCA, EDBF, RCBBFA and ABPSO on all benchmark functions in terms of efficiency.

6. Conclusion

In this paper, we proposed an enhanced EDBF framework, namely CLM, by adaptively changing the boundary of weight assigned to each parent chromosome according to the constraint law. Compared with EDBF that spawn weights in different ranges bounded by the empirical probability, CLM spawn weights in the adaptive range, and just once loop the weights satisfying the boundrary could be spawned. Furthermore, CLM is compared with EDBF, ABPSO and RCBBFA algorithms on 20 benchmark functions in terms of convergence, efficiency and accuracy. Experimental results demonstrate that CLM outperforms comparative algorithms on most of benchmark functions. As a general framework rather than a specific algorithm, CLM is easy to implement and can easily be accommodated to any existing MCAs. Finally, the C++ source code is available at https://github.com/ZhengkangZUO-2020/CLM-Framework-Codes.

Conflict of Interest

The authors declare that there is no conflict of interests regarding the publication of article.

Table 5. Average efficiency of CLM, EMCA, EDBF, RCBBFA and ABPSO for 20 benchmark functions.

Benchmark functions	Average run time after 100 independent runs (seconds)				
	EMCA+CLM	EMCA	EMCA+EDBF	RCBBFA	ABPSO
f_1	**0.688**	27.517	4.043	5.009	1.695
f_2	**1.247**	53.347	7.467	9.382	2.259
f_3	**0.697**	29.371	4.260	7.619	8.319
f_4	**1.185**	60.097	8.324	13.698	11.360
f_5	**0.889**	31.117	4.351	6.645	7.471
f_6	**0.226**	7.800	1.121	6.112	7.374
f_7	**0.254**	11.603	1.710	16.795	13.842
f_8	**0.322**	7.689	0.790	4.276	2.788
f_9	**0.302**	5.813	0.572	3.303	2.096
f_{10}	**0.950**	40.600	5.620	10.840	11.504
f_{11}	**1.128**	26.138	3.469	4.030	2.751
f_{12}	**0.606**	25.663	3.645	18.722	15.037
f_{13}	**0.607**	25.751	3.733	17.335	14.185
f_{14}	**0.110**	1.234	0.233	33.052	11.202
f_{15}	**0.323**	7.434	0.921	5.409	4.297
f_{16}	**0.138**	3.600	1.284	5.858	3.394
f_{17}	**0.160**	16.779	11.188	3.544	2.342
f_{18}	**1.871**	29.710	3.096	4.752	2.978
f_{19}	**0.701**	26.451	4.435	100.387	67.962
f_{20}	**0.402**	18.558	2.742	6.754	7.842

Acknowledgments

I am appreciated for useful comments from professor Zhijian Wu.

References

[1] Harik, G. R., Lobo, F. G., and Goldberg, D. E. (1999). The compact genetic algorithm. *IEEE transactions on evolutionary computation*, *3*(4), 287-297. https://doi.org/10.1109/4235.797971.
[2] Jiang, D., and Lin, J. (2010). A unified multi-parent combination algorithm. *Journal of Huazhong University of Science and Technology* (Natural Science Edition), 12. (In Chinese)
[3] Wang, F., et al. A hybrid particle swarm optimization algorithm using adaptive learning strategy. Information Sciences, 2018, 436: 162-177. https://doi.org/10.1016/j.ins.2018.01.027.
[4] Tan, Z., Li, K., and Wang, Y. Differential evolution with adaptive mutation strategy based on fitness landscape analysis. Information Sciences, 2021, 549: 142-163. https://doi.org/10.1016/j.ins.2020.11.023.
[5] Zuo, Z., Yan, L., Ullah, S., Sun, Y., Zhang, R., and Zhao, H. (2021). Empirical distribution-based framework for improving multi-parent crossover algorithms. Soft Computing, 25(6), 4799-4822. https://doi.org/10.1007/s00500-020-05488-1.
[6] Zuo, Z., Sun, Y., Sun, J., Zhang, R., and Yan, L. (2020). Accelerating the generation of coefficient vectors in elite multi-parent crossover algorithm by using empirical probability density curve. *Engineering Journal of Wuhan University*, 53(08):728-733. https://doi.org/10.14188/j.1671-8844.2020-08-011. (In Chinese)
[7] Wu Z, Kang L, and Zou X. (2003). An elite subspace evolutionary algorithm for solving function optimization problems. *Journal of Computer Applications*, 23(2): 13-15. (In Chinese)
[8] Wu, Z., Kang, L., and Zou, X. (2002). A parallel global-local mixed evolutionary algorithm for multimodal function optimization. In: *Fifth International Conference on Algorithms and Architectures for Parallel Processing, 2002. Proceedings*, 247-250. https://doi.org/10.1109/ICAPP.2002.1173582.

[9] Wu, Z., Tang, Z., and Kang, L. (2003). A parallel global-local mixed evolutionary algorithm for multimodal function optimization based on domain decomposition. *Wuhan University Journal of Natural Sciences*, 8(1), 253-258. https://doi.org/10.1007/BF02899489.
[10] Zuo, Z., et al., (2021). Trajectory Simulation and Prediction of COVID-19 via Compound Natural Factor (CNF) Model in EDBF Algorithm. *Earth's Future*, e2020EF001936. https://doi.org/10.1029/2020EF001936.
[11] Ullah, S., et al., (2020). GPM-Based Multitemporal Weighted Precipitation Analysis Using GPM_IMERGDF Product and ASTER DEM in EDBF Algorithm. *Remote Sensing*, 12(19), 3162. https://doi.org/10.3390/rs12193162.
[12] Zuo, Z., et al. (2020). CMIP5 climate multi-model ensemble optimization based on tempo-spatially distributed. *Acta Scientiarum Naturalium Universitatis Pekinensis*, 56(05):805-814. https://doi.org/10.13209/j.0479-8023.2020.057. (In Chinese)
[13] Zuo., Z., et al. (2020). Improved Genetic Algorithm for Bundle Adjustment in Photogrammetry. IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium, 6957-6960. https://doi.org/10.1109/IGARSS39084.2020.9323649.
[14] Zhang, Y., Gong, D. W., Sun, X. Y., and Geng, N. (2014). Adaptive bare-bones particle swarm optimization algorithm and its convergence analysis. Soft Computing, 18(7), 1337-1352. https://doi.org/10.1007/s00500-013-1147-y.
[15] Zhang, Y., Song, X. F., and Gong, D. W. (2017). A return-cost-based binary firefly algorithm for feature selection. Information Sciences, 418, 561-574. https://doi.org/10.1016/j.ins.2017.08.047.
[16] Yao, X., Liu, Y., and Lin, G. (1999). Evolutionary programming made faster. IEEE Transactions on Evolutionary computation, 3(2), 82-102. https://doi.org/10.1109/4235.771163.