Cooperative coevolutionary differential evolution with improved augmented Lagrangian to solve constrained optimisation problems

Ghasemishabankareh, Behrooz; Li, Xiaodong; Ozlen, Melih

Ghasemishabankareh, B., Li, X., & Ozlen, M. (2016). Cooperative coevolutionary differential evolution with improved augmented Lagrangian to solve constrained optimisation problems. Information Sciences, 369, 441–456. https://doi.org/10.1016/j.ins.2016.06.047
Document Version: Accepted Manuscript

Published Version: https://doi.org/10.1016/j.ins.2016.06.047

Cooperative coevolutionary differential evolution with improved augmented Lagrangian to solve constrained optimisation problems

Behrooz Ghasemishabankareh*, Xiaodong Li, Melih Ozlen

School of Science, RMIT University, Melbourne, Australia

Abstract

In constrained optimisation, the augmented Lagrangian method is considered as one of the most effective and efficient methods. This paper studies the behaviour of augmented Lagrangian function (ALF) in the solution space and then proposes an improved augmented Lagrangian method. We have shown that our proposed method can overcome some of the drawbacks of the conventional augmented Lagrangian method. With the improved augmented Lagrangian approach, this paper then proposes a cooperative coevolutionary differential evolution algorithm for solving constrained optimisation problems. The proposed algorithm is evaluated on a set of 24 well-known benchmark functions and five practical engineering problems. Experimental results demonstrate that the proposed algorithm outperforms the state-of-the-art algorithms with respect to solution quality as well as efficiency.

Keywords: Augmented Lagrangian method, Constrained optimisation, Cooperative coevolution, Differential evolution

1. Introduction

Most of the science and engineering optimisation problems in real world are highly constrained. These constrained optimisation problems present se-

*Corresponding author

Email addresses: behrooz.ghasemishabankareh@rmit.edu.au (Behrooz Ghasemishabankareh), xiaodong.li@rmit.edu.au (Xiaodong Li), melih.ozlen@rmit.edu.au (Melih Ozlen)

rious challenges to existing optimisation methods. Developing effective constraint handling techniques is critical in addressing these challenges. Constraint handling techniques can be categorised into four groups [26]: maintaining feasibility of solutions, penalty functions, distinguishing between feasible and infeasible solutions and hybrid methods. Other categorisations are also possible [8]. Generally speaking, each of these techniques has some advantages and disadvantages.

Evolutionary algorithms (EAs) have been applied to various optimisation problems which classical optimisation algorithms cannot be directly applied to or do not provide promising results [40]. One of the most common constraint handling techniques with EA is the penalty function approach, as presented by Courant et al. [10]. The penalty function approach converts a constrained optimisation problem into a sequence of unconstrained problems by adding a penalty term to the original objective function to penalise infeasible solutions [28]. However penalty functions are often not differentiable and this is the main drawback of using this approach. Another popular approach is the Lagrangian multiplier method, which is based on Kuhn-Tucker conditions and can be used to convert a constrained optimisation problem into an unconstrained one. However, this approach assumes the problem to be convex. In order to handle non-convex problems, the augmented Lagrangian method is introduced in [38], to convexify the objective function by adding quadratic penalty terms [44].

Many studies have been carried out during the last decade to solve constrained optimisation problems using the augmented Lagrangian approach. Adeli and Cheng [1] proposed a hybrid genetic algorithm (GA) to solve structural optimisation using ALF. Sarma and Adeli [42] presented a fuzzy augmented Lagrangian method using GA to optimise steel structures. Rocha et al. [37] proposed a stochastic population based algorithm using the augmented Lagrangian method. An artificial fish swarm algorithm based hyperbolic augmented Lagrangian method is used to solve constrained optimisation problems in [9]. An ant colony optimisation (ACO) algorithm with augmented Lagrangian method is presented in [21] to solve continuous global optimisation problems. Mallipeddi and Suganthan [22] presented an ensemble of four different constraint handling methods to solve constrained optimisation problems.

Dealing with complex combinatorial solution spaces as well as problems with high number of constraints is a challenging task. Some attempts have been made to hybridise the EAs with local search algorithms to cope with

this challenge in an efficient way [40] e.g., a hybrid particle swarm optimisation(PSO) with GA [14]. Another approach to deal with the aforementioned challenge is using a coevolutionary algorithm. Tahk and Sun [44] presented a coevolutionary algorithm using zero-sum game to coevolve the decision variables and Lagrangian multipliers. A coevolutionary GA has been also used to solve constrained optimisation problems [3]. Krohling et al. [19] used a coevolutionary PSO augmented Lagrangian function to deal with constraints. They proposed a Gaussian probability distribution for the acceleration coefficient in PSO. Nema et al. [29] presented a hybrid coevolutionary algorithm with the min-max approach to solve constrained optimisation problems. They also used an augmented Lagrangian method to handle constraints.

Although ALF is an efficient method to deal with constraints, it changes the fitness values dramatically for solutions lying far from the boundaries of the feasible space. In this paper, we propose an improved augmented Lagrangian function (iALF) to handle this issue in a more effective manner. Based on iALF, we also propose an efficient CCiALF method for solving constrained optimisation problems. The proposed algorithm produces higher quality solutions using fewer number of function evaluations (NFE). To demonstrate the capability of CCiALF algorithm, two sets of benchmarks are used in our study and the results are compared with that of the state-of-the-art algorithms.

The rest of the paper is structured as follows: First some background on ALF is described in section 2, and then our improved ALF (iALF) is presented in section 3. The proposed CCiALF method is introduced in Section 4 and experimental results are presented in Section 5. Finally, Section 6 provides conclusion and future research directions.

2. Augmented Lagrangian function

The general constrained optimisation problem can be described as follows:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \qquad \boldsymbol{x} \in R^p \tag{1}$$

$$g_i(\boldsymbol{x}) \geq 0, \qquad i = 1, ..., m \tag{2}$$

$$h_j(\boldsymbol{x}) = 0, \qquad j = 1, ..., n \tag{3}$$

$$lb_k \leq x_k \leq ub_k, \;\; k = 1, ..., p \tag{4}$$

where Eq. (1) represents the objective function, Eqs. (2) and (3) are inequality and equality constraints, respectively. Eq. (4) represents lower and upper bounds on decision variables $\boldsymbol{x}$. In [16] and [31], only the equality constraints are considered and the above problem is transformed into an unconstrained one by adding quadratic penalty terms and dual values to the objective function. Rockafellar [39] utilised the idea and modified it for inequality constraints. The augmented Lagrangian (also called as penalty Lagrangian by Rockafellar [38]) replacing the quadratic penalty term by θ function is presented as follows [13]:

$$F(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\tau}) = f(\boldsymbol{x}) + R \sum_{j=1}^{m} [(\theta(g_j(\boldsymbol{x}) + \mu_j))^2 - (\mu_j)^2] + R \sum_{k=1}^{n} [(h_k(\boldsymbol{x}) + \tau_k)^2 - (\tau_k)^2] \tag{5}$$

$$\theta(G) = min\{0, G\} \tag{6}$$

where R is a positive penalty parameter, $\boldsymbol{\mu}$ is a $1 \times m$ multiplier, $\boldsymbol{\tau}$ is a $1 \times n$ multiplier for inequality and equality constraints respectively, and G can be any function or value. The θ function checks if the inner expression (e.g. G) is greater than zero or not. If $G \geq 0$ then $\theta(G) = 0$, otherwise $\theta(G) = G$.

Deb and Srivastava [13] evaluated the classical ALF using benchmark functions and it is shown that the classical ALF has limited success, e.g., the solutions are still far from the known optima. Fig. 1 shows an example of a 2-dimensional problem where ALF divides the search space into four different regions: the inner feasible region, ie., the inner feasible area far from the boundary (region 1), the feasible area close to the boundary (region 2), the feasible area on the boundary(region 3) and finally the infeasible area (region 4).

For the sake of simplicity, we consider the optimisation problems with only inequality constraints. Now the ALF is formulated as follows:

$$F(\boldsymbol{x}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + R \sum_{j=1}^{m} [(\theta(g_j(\boldsymbol{x}) + \mu_j))^2 - (\mu_j)^2] \tag{7}$$

If we consider the problem has one constraint (m=1), the ALF value in the feasible and infeasible regions are as follows:

- **Region 1**: For a given point $\boldsymbol{w}$ (in region 1), since $\mu < 0$, $|\mu| < g(\boldsymbol{w})$, $G = (g(\boldsymbol{w}) + \mu)$ is greater than zero and according to Eq. (6), $\theta(g(\boldsymbol{w}) + \mu) = 0$. By substitution of the values in Eq. (7), $F = f(\boldsymbol{w}) - R\mu^2$.

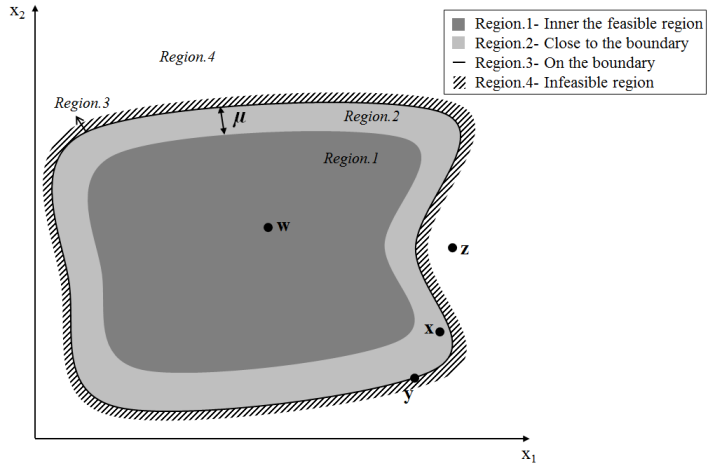


Figure 1: Four different regions divided by ALF.

- **Region 2**: For a given point $\boldsymbol{x}$ (in region 2), since $\boldsymbol{x}$ is close to the boundary and $\mu < 0$, $|\mu| > g(\boldsymbol{x})$, then $\theta(g(\boldsymbol{x}) + \mu) = g(\boldsymbol{x}) + \mu$ and ALF value is $F = f(\boldsymbol{x}) + R(g(\boldsymbol{x})^2 + 2g(\boldsymbol{x})\mu)$.

- **Region 3**: For a given point $\boldsymbol{y}$ (in region 3), $\mu < 0$ and $g(\boldsymbol{y}) = 0$. As a result, $\theta(g(\boldsymbol{y}) + \mu) = \mu$ and $F = f(\boldsymbol{y})$.

- **Region 4**: For a given point $\boldsymbol{z}$ (in region 4), $\mu < 0$, $g(\boldsymbol{z}) < 0$, $\theta(g(\boldsymbol{z}) + \mu) = g(\boldsymbol{z}) + \mu$ and the ALF value is $F = f(\boldsymbol{z}) + R(g(\boldsymbol{z})^2 + 2g(\boldsymbol{z})\mu)$.

Table 1 summarises the calculation procedure of the augmented Lagrangian function in different regions. The fifth column in Table 1 demonstrates the ALF values for all points $\boldsymbol{w}$, $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ in regions 1, 2, 3 and 4 respectively. Note that we assume R has a large positive value.

Since R is a large positive number and μ has a negative value, for region 1, $R\mu^2$ is a large positive number and by subtracting $R\mu^2$ from the objective function value f, the ALF value F decreases dramatically. In region 2, since a solution is close to the boundary ($g > 0$), g^2 is a small positive number and $2g\mu$ is a negative number where ($|g^2| < |2g\mu|$). So by adding $R(g^2 + 2g\mu)$ to the f, the value of F decreases slightly. In region 3, F is equal to the original objective function value f. Finally in region 4, a solution is infeasible ($g < 0$), g^2 and $2g\mu$ are both positive numbers. Hence, by adding the penalty value of $R(g^2 + 2g\mu)$ to f, the value of F increases. Adding an extra value to

the original objective function value in regions 1 and 2 changes the fitness landscape. To illustrate this problem, an example is presented here.

Consider problem g24 from CEC'2006 [20]. The problem formulation is as follows:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) = -x_1 - x_2 \\ \text{s.t.} \quad & g_1(\boldsymbol{x}) = 2x_1^4 - 8x_1^3 + 8x_1^2 - x_2 + 2 \geq 0 \\ & g_2(\boldsymbol{x}) = 4x_1^4 - 32x_1^3 + 88x_1^2 - 96x_1 - x_2 + 36 \geq 0 \\ & 0 \leq x_1 \leq 3, \quad 2 \leq x_2 \leq 4 \end{aligned} \tag{8}$$

In this problem two points ($\boldsymbol{x_A}$ and $\boldsymbol{x_B}$) are selected from regions 1 and 2, respectively. Assume that $\mu = -1$ and $R = 50000$ (a positive large penalty value). As shown in Table 2, $\boldsymbol{x_A}$ and $\boldsymbol{x_B}$ are both feasible solutions satisfying all constraints and since $f(\boldsymbol{x_A}) \geq f(\boldsymbol{x_B})$, $\boldsymbol{x_B}$ is better than $\boldsymbol{x_A}$. Although $\boldsymbol{x_B}$ is better than $\boldsymbol{x_A}$, the ALF value for $\boldsymbol{x_A}$ and $\boldsymbol{x_B}$ shows that $F(\boldsymbol{x_A}) \leq F(\boldsymbol{x_B})$ incorrectly suggesting that $\boldsymbol{x_A}$ is better than $\boldsymbol{x_B}$.

The above-mentioned problem for ALF occurs because ALF drastically distorts the fitness landscape of the feasible areas in regions 1 and 2. These changes in fitness values may cause ALF to give misleading information. In this case, the solution fitness value in region 1 is dramatically decreased, whereas the fitness of solution in region 2 is only slightly decreased, giving an incorrect result.

3. Improved augmented Lagrangian function

In this section the improved ALF (iALF) is presented. We have shown in the previous section that ALF has some issues in regions 1 and 2 of the search space. The main issue is that ALF distorts the fitness values for feasible solutions. It would be desirable to minimise the amount of distortion of fitness in the feasible region. Ideally, two feasible solutions should just be compared according to the original objective function values and an extra value should not be added. However, in ALF the extra value distorts the landscape of fitness in the feasible region. To rectify this issue, we propose an iALF as follows:

$$iF(\boldsymbol{x}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + R \sum_{j=1}^{m} [(\theta(g_j(\boldsymbol{x})) + \mu_j)^2 - (\mu_j)^2] \tag{9}$$

where if point $\boldsymbol{x}$ is a feasible solution (located in region 1, 2 or 3) the iALF value iF equals to the original objective value f and if point $\boldsymbol{x}$ is an infeasible solution (located in region 4) iF is set to equal to $f+R\sum(g_j(\boldsymbol{x})^2+2g_j(\boldsymbol{x})\mu_j)$. In other words, if $\boldsymbol{x}$ is in the feasible area, iALF should not change the fitness landscape and objective function value will stay the same as its iF value. Whereas, if $\boldsymbol{x}$ is in the infeasible area, the $R\sum(g_j(\boldsymbol{x})^2+2g_j(\boldsymbol{x})\mu_j)$ term is added to the objective function value to penalise the infeasible solution. The last column of Table 1 shows the iF values in different regions.

Table 1: F vs. iF values in different regions.

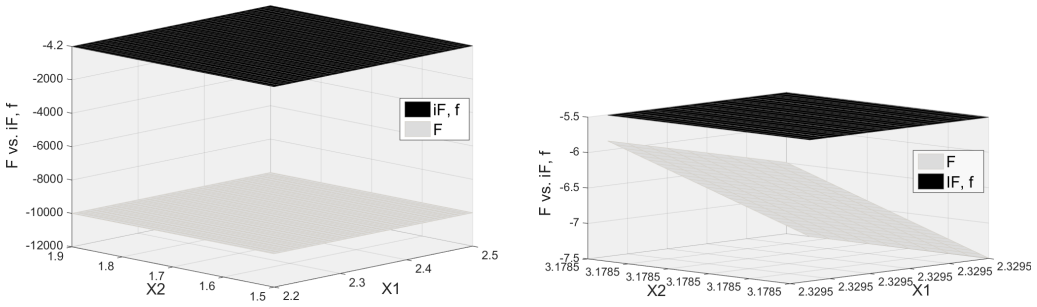
Points	g($\boldsymbol{x}$)	$\boldsymbol{\mu}$	$\theta(g+\mu)$	F	iF
$\boldsymbol{w}$ (Region 1)	>0	<0	0	$f(\boldsymbol{w})-R\mu^2$	$f(\boldsymbol{w})$
$\boldsymbol{x}$ (Region 2)	>0	<0	$g(\boldsymbol{x})+\mu$	$f(\boldsymbol{x})+R(g(\boldsymbol{x})^2+2g(\boldsymbol{x})\mu)$	$f(\boldsymbol{x})$
$\boldsymbol{y}$ (Region 3)	$=0$	<0	μ	$f(\boldsymbol{y})$	$f(\boldsymbol{y})$
$\boldsymbol{z}$ (Region 4)	<0	<0	$g(\boldsymbol{z})+\mu$	$f(\boldsymbol{z})+R(g(\boldsymbol{z})^2+2g(\boldsymbol{z})\mu)$	$f+R(g(\boldsymbol{z})^2+2g(\boldsymbol{z})\mu)$

The iALF is applied to calculate the fitness function for problem g24 in Table 2. The last row presents the iF value for points $\boldsymbol{x_A}$ and $\boldsymbol{x_B}$. As discussed earlier, since both points are feasible, iALF will not change the fitness landscape. Hence, the iF value is equal to the objective function value f. Since $iF(\boldsymbol{x_A}) \geq iF(\boldsymbol{x_B})$ which is the same as $f(\boldsymbol{x_A}) \geq f(\boldsymbol{x_B})$, iALF will give us the correct result, i.e., $\boldsymbol{x_B}$ is better than $\boldsymbol{x_A}$.

Table 2: F and iF for problem g24.

Point $\boldsymbol{x_A}$		Point $\boldsymbol{x_B}$	
x_1	2.404678	x_1	2.329524
x_2	1.797169	x_2	3.178475
$g_1(\boldsymbol{x_A})$	$2.096757 \geq 0$	$g_1(\boldsymbol{x_B})$	$4.91E-05 \geq 0$
$g_2(\boldsymbol{x_A})$	$0.999992 \geq 0$	$g_2(\boldsymbol{x_B})$	$2.03E-07 \geq 0$
$f(\boldsymbol{x_A})$	**-4.201847**	$f(\boldsymbol{x_B})$	**-5.507999**
μ_1	-1	μ_1	-1
μ_2	-1	μ_2	-1
R	50000	R	50000
$F(\boldsymbol{x_A})$	**-100004.2018**	$F(\boldsymbol{x_B})$	**-10.440193**
$iF(\boldsymbol{x_A})$	-4.201847	$iF(\boldsymbol{x_B})$	-5.508012

Fig. 2 demonstrates the distortion of the fitness function in neighbourhood of points $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ by ALF. As shown in Fig. 2, the iF and f values



(a) F, iF and f in region 1 (point $\boldsymbol{x}_A$) (b) F, iF and f in region 2 (point $\boldsymbol{x}_B$)

Figure 2: Comparison of ALF and iALF in different regions for problem g24.

are equal while the F value is dramatically distorted in Fig. 2a and slightly distorted in Fig. 2b.

4. The proposed CCiALF algorithm

In our proposed CCiALF method, iALF is used to handle constraints and differential evolution (DE) is adopted as an subcomponent optimiser within the cooperative coevolutionary framework. CCiALF algorithm coevolves two populations which interact with each other at each iteration. According to Eq. (9), the iALF has two sets of variables ($\boldsymbol{x}$ and $\boldsymbol{\mu}$). The first population (PopulationI) evolves the decision vector $\boldsymbol{x}$, whereas the second population (PopulationII) evolves the multiplier vector $\boldsymbol{\mu}$. The fitness of an individual in one population is evaluated according to how well it collaborates with the best-fit individual from the other population. The procedure of DE, the cooperative coevolutionary method and CCiALF algorithm are described in the following subsections.

4.1. DE algorithm

DE is a powerful stochastic population-based optimisation method which is proposed by Storn and Price [43]. DE evolves a population of N_{pop} p-dimensional vectors in generation It, i.e., $\boldsymbol{x}_{i,It} = (x^1_{i,It}, \ldots, x^p_{i,It})$, $i = 1, \ldots, N_{pop}$ from one generation to the next. The initial population can be generated as follows:

$$\boldsymbol{x}_{i,1} = \boldsymbol{lb} + rand(0,1).(\boldsymbol{ub} - \boldsymbol{lb}); i = 1, 2, \cdots, N_{pop} \tag{10}$$

where $rand(0,1)$ is an uniform random variable in [0,1] and $\boldsymbol{ub}$, $\boldsymbol{lb}$ are upper and lower bound vectors for $\boldsymbol{x}$, respectively.

After initialisation phase, a mutation operator is performed on the target vector $\boldsymbol{x}_{i,It}$ to generate a mutant vector $\boldsymbol{v}_{i,It}$. There are different strategies to produce a mutant vector [32]. After mutant vectors are generated, the crossover operator should be performed on $\boldsymbol{x}_{i,It}$ and its corresponding mutant vector $\boldsymbol{v}_{i,It}$ to produce a trial vector ($\boldsymbol{u}_{i,It}$). Binomial and exponential crossover techniques are two general crossover operators.

If any of the generated trial vectors does not satisfy Eq. (4), the trial vector should be regenerated randomly in [$\boldsymbol{lb}$,$\boldsymbol{ub}$]. Then the fitness function of the trial vector $\boldsymbol{u}_{i,It}$ and the corresponding target vector $\boldsymbol{x}_{i,It}$ should be compared and the next generation should be formed. The binary selection procedure is as follows:

$$\boldsymbol{x}_{i,It+1} = \begin{cases} \boldsymbol{u}_{i,It}, & if f(\boldsymbol{u}_{i,It}) \leq f(\boldsymbol{x}_{i,It}) \\ \boldsymbol{x}_{i,It}, & otherwise \end{cases}$$

The aforementioned steps are performed until the termination condition is satisfied. Since the performance of DE algorithm significantly depends on adopting trial vector generation strategies and their associated control parameter values, numerous variants of DE algorithm have been introduced during the last decade. Qin and Suganthan [33] proposed a self adaptive DE where the choice of learning strategy and two control parameters are not required to be pre-specified by a user. Brest et al. [6] presented a new version of DE (jDE) for obtaining self adaptive control parameter settings. In order to improve the optimisation performance, Zhang and Sanderson [46] introduced a novel DE algorithm (JADE) by applying a new mutation strategy with an optional external archive and updating control parameters in an adaptive manner. A multi-criteria adaptive DE (MADE) algorithm utilises a multi-criteria adaptation approach to choose trail vector generation strategies and separately adjusts the control parameters of each strategy [7]. A self adaptive DE (SaDE) is presented in [32] to make the trial vector generation strategies and their associated control parameter values self-adapted according to the experiences at previous generations. Other variations of DE algorithm with ensemble of parameters and strategies are proposed in [18] and [17].

4.2. Cooperative coevolutionary algorithm

Fig. 3 shows how a cooperative coevolutionary algorithm works. As mentioned before, iALF has two variables: $\boldsymbol{x}$, the decision variable vector for the problem to be optimised, and $\boldsymbol{\mu}$, the Lagrangian multipliers. PopulationI

and PopulationII evolve $\boldsymbol{x}$ and $\boldsymbol{\mu}$, respectively. In Fig. 3, m, n, p, N_{pop_1}, N_{pop_2} are the number of inequalities, the number of equalities, the number of decision variables, the number of individuals in PopulationI and the number of individuals in PopulationII, respectively.

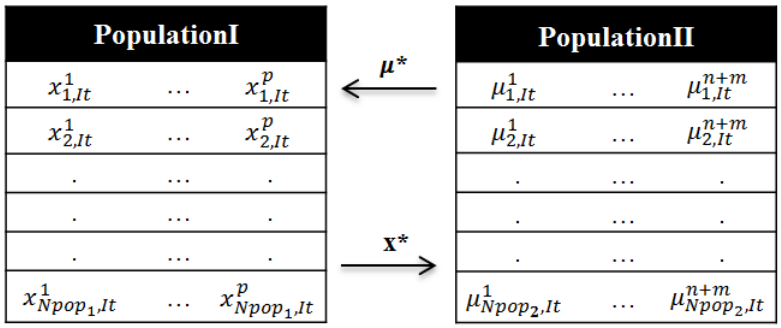


Figure 3: Cooperative coevolutionary algorithm.

PopulationI ($\boldsymbol{x}$) and PopulationII ($\boldsymbol{\mu}$) should be initialised randomly according to Eq. (10). Note that for populationII, $\boldsymbol{lb}$, $\boldsymbol{ub}$ are $\boldsymbol{\mu_0}$ and $\mathbf{0}$, respectively. In order to evaluate the fitness function for PopulationI and II, members of both populations should interact and $\boldsymbol{x}$, $\boldsymbol{\mu}$ should be given. Therefore in PopulationI, at the initialisation step, each individual is combined with a random individual from Population II, but from the next iteration onwards, each individual ($\boldsymbol{x}_{i,It}; i = 1, \ldots, Npop_1$) is combined with the best $\boldsymbol{\mu}$ vector ($\boldsymbol{\mu}^*$), and this combined vector is evaluated using the fitness function (Eq. (9)) to assess how well the individual collaborates with the best individual from the other population. Similarly, to evaluate each individual in PopulationII, vector ($\boldsymbol{\mu}_{j,It}; j = 1, \ldots, Npop_2$) is combined with the best individual in PopulationI ($\boldsymbol{x}^*$). Then Eq. (9) is applied to calculate the fitness function values for all members in PopulationII. At each iteration (It), each individual in PopulationI and Population II is coevolved in a round-robin fashion [30]. The following procedure continues until the termination condition (e.g. maximum number of iterations (It_{max})) is satisfied and the $\boldsymbol{x}^*$ is reported as algorithm's final solution.

4.3. CCiALF algorithm

Algorithm 1 provides the pseudo-code describing the proposed CCiALF algorithm in detail. PopulationI and II are randomly generated in the range [$\boldsymbol{lb}$,$\boldsymbol{ub}$]. Then by combining two subpopulations, the fitness function value

is calculated and the best individual from each sub-population ($\boldsymbol{x}^*$,$\boldsymbol{\mu}^*$) is selected. Since the performance of DE highly depends on the efficient trial vector generation strategies and associated control parameter values, the SaDE is adopted to make the trial vector generation strategies and their associated control parameter values self-adaptive according to the experiences from previous generations [32]. During the iterations of CCiALF, SaDE [32] is used as a subcomponent optimiser to evolve PopulationI and II. As shown in Eq .(9) the penalty parameter R is needed to evaluate the iALF value. The adaptive procedure for R value provides a balance between objective function value and constraint violation [13]. Consequently, the R parameter is updated according to the statistics of the current population. Hence, in the CCiALF procedure, in every β_2 iterations, R is updated according to the method presented in [13].

In addition, to improve the convergence speed, in every β_1 iterations, if the difference between the best objective value in two successive iterations is less than δ, which is defined by a user, a point-based local search is performed on the worse solution in the current generation. Using a local search procedure within CCiALF algorithm, increases the convergence speed and prevents CCiALF from getting trapped at local optima. To save computational cost, the local search procedure is just performed on PopulationI.

For the local search procedure, any point-based optimisation algorithm can be used. Similar to [13], in this paper $fmincons()$ routine of MATLAB is used [12]. $fmincons()$ attempts to find the minimum of a scalar function of several variables starting at an initial estimate. This function uses an interior point method by default to solve unconstrained problems [23].

The termination criterion is the maximum NFE (MaxNFE) or no improvement in the objective function value in β_3 successive iterations. The algorithm stops when it reaches MaxNFE or it has not improved for β_3 number of iterations.

5. Experimental results

We use a well-known constrained optimisation test functions suite, CEC'2006 [20] for evaluating CCiALF algorithm. In addition, five practical engineering problems from [34],[41],[4],[15] and [35] are also used to assess the CCiALF's performance on solving real-world engineering problems. In order to find out how well the iALF performs against the conventional ALF method, the performance of iALF is first compared with ALF. Then, we apply CCiALF

Algorithm 1 CCiALF algorithm

procedure $CCiALF(N_{pop_1}, N_{pop_2}, It_{max}, \mu_0, \beta_1, \beta_2, \delta, Const, MaxNFE)$
  Initialisation
  Randomly generate $PopulationI_{,0}$ and $PopulationII_{,0}$ (Eq. (10))
  Initialise R_0 as follows :$R_0 = \frac{\sum_{i=1}^{N_{pop_1}} |f_i(\boldsymbol{x})|}{\sum_{i=1}^{N_{pop_1}} \sum_{j=1}^{m+n} \theta(g_{ij}(\boldsymbol{x}))}$; $R_{old} = R_0$
  Calculate $iF_{1,0}(\boldsymbol{x}, \boldsymbol{\mu})$, $iF_{2,0}(\boldsymbol{x^*}, \boldsymbol{\mu})$ for PopulationI,II respectively (Eq. (9))
  Add NFE_0
  $BFS(1) = \boldsymbol{x}^*$
  Select $(\boldsymbol{x}^*, \boldsymbol{\mu}^*)$
  Main Loop
  for all $It = 1$: It_{max} **do**
    for all s= I: II **do** % Number of subpopulations%
      Perform SaDE on $Population_{s,It}$
      Combine the $Population_{s,It}$ with the best individual of other subpopulation
      Calculate fitness functions for combined individuals
      $Population_{s,It+1} \leftarrow$ SaDE Selection Procedure on $Population_{s,It}$
      Add NFE_0
    end for
    if $mod(It_{Max}, \beta_1) = 0$ **and** $|BFS(It) - BFS(It-1)| < \delta$ **then**
      Do local search on the worst vector in PopulationI and add NFE_0
    end if
    if $mod(It_{Max}, \beta_2) = 0$ **then**
      Calculate R using: $R = 0.5\frac{f(\boldsymbol{x}^*)}{N_{pop_1}} + 0.5R_{old}$
    else
      $R = R_{old}$
    end if
    Select $(\boldsymbol{x}^*, \boldsymbol{\mu}^*)$
    $BFS(It+1) = \boldsymbol{x}^*$
    if $|BFS(It) - BFS(It-1)| = 0$ **then**
      $count = count + 1$
    else
      $count = 0$
    end if
    if $NFE_0 > MaxNFE$ **or** $count > \beta_3$ **then**
      break
    end if
  end for
end procedure

(Algorithm 1) to the benchmark functions as well as several engineering problems to demonstrate the capability of CCiALF algorithm.

The proposed CCiALF algorithm is implemented in MATLAB on a PC with Intel(R) Core(TM) i7-4790 3.60 GHz processor with 8 GB RAM and run 30 times for each set of benchmark. Parameter settings of algorithm are as follows: $N_{pop_1} = 200$, $N_{pop_2} = 20$, $It_{Max} = 500$, $\mu_0 = -5$, $\beta_1 = 0.1 * It_{Max}$, $\beta_2 = 5$, $\delta = 0.0001$, $\beta_3 = 0.02 * It_{Max}$, $MaxNFE = 240000$.

5.1. ALF vs. iALF

In this section CEC'2006 benchmarks are used to evaluate both ALF and iALF to demonstrate the performance difference between these two methods. As mentioned in Section 2, ALF changes the fitness landscape of the solution space and it may affect the performance of the algorithm dramatically. iALF described in Section 3 aims to combat this issue. If Eq. (9) is replaced by Eq. (7) in CCiALF (Algorithm 1) and used to evaluate the fitness function value, this algorithm is called as CCALF algorithm. We evaluate both CCALF and CCiALF on the CEC'2006 benchmarks. To help understand better whether the improvement comes from this iALF, we deliberately remove the local search from this comparison. We adopt a CCiALF without local search, called $\text{CCiALF}_{\text{NLS}}$, which is compared directly with $\text{CCALF}_{\text{NLS}}$.

As shown in Eqs. (1), (2) and (3) the constrained problems are composed of equality and inequality constraints. The equality constraints Eq. (2) are transformed into inequality constraints by Eq. (11) [20].

$$|h_i(\boldsymbol{x})| - \epsilon \leq 0 \tag{11}$$

where ϵ is a tolerance parameter for equality constraints. The value of ϵ is equal to 0.0001. By applying above-mentioned transformation, the problem is reformulated as follows:

$$\begin{aligned} &\min_{\boldsymbol{x}} f(\boldsymbol{x}), && \boldsymbol{x} \in R^n \\ &g_i(\boldsymbol{x}) \geq 0, && i = 1, ..., m \\ &h_i(\boldsymbol{x}) + \epsilon \geq 0, && i = 1, ..., n \\ &\epsilon - h_i(\boldsymbol{x}) \geq 0, && i = 1, ..., n \\ &lb_i \leq x_i \leq ub_i, && i = 1, ..., p \end{aligned}$$

The CEC'2006 test functions are used to evaluate the performance of $\text{CCALF}_{\text{NLS}}$ and $\text{CCiALF}_{\text{NLS}}$ algorithms. For each problem 30 independent

runs are performed and mean (Mn) and standard deviation (sd) are provided in Table 3. In order to compare the performance of CCiALF$_{\text{NLS}}$ with CCALF$_{\text{NLS}}$ algorithm, the t-test with the significance level of 0.05 is performed. To perform the t-test the following hypotheses are considered.

$$\begin{aligned} H_0 &: Mn_j = Mn_i \\ H_a &: Mn_j \neq Mn_i \end{aligned} \tag{12}$$

where j=CCiALF$_{\text{NLS}}$ and i=CCALF$_{\text{NLS}}$. After performing the pairwise t-test for both algorithms, if CCiALF$_{\text{NLS}}$ is equal, superior or inferior to the compared CCALF$_{\text{NLS}}$ algorithm, then h is equal to 0, 1 and -1 respectively. As shown in Table 3, CCiALF$_{\text{NLS}}$ is superior to CCALF$_{\text{NLS}}$ in 18 test functions (out of 22) and in 2 test functions both algorithms have the same performance. In addition, CCALF$_{\text{NLS}}$ failed to obtain any feasible solution for g03 and g14 (denoted by "inf" in Table 3). Fig. 4 shows the convergence plots for CCiALF$_{\text{NLS}}$ and CCALF$_{\text{NLS}}$ algorithms in solving test problems g08, g12, g16 and g19. These problems are chosen because g08 and g12 require low computational cost (approximately 20,000 NFE). On the other hand, g16 and g19 use high computational cost (more than 200,000 NFE). As shown in Fig. 4, CCiALF$_{\text{NLS}}$ converged to better solutions and were faster than CCALF$_{\text{NLS}}$. The aforementioned results demonstrate that CCiALF$_{\text{NLS}}$ outperforms CCALF$_{\text{NLS}}$ in terms of constraint handling effectiveness, solution quality and convergence speed.

Table 3: Comparison between CCiALF$_{\text{NLS}}$ and CCALF$_{\text{NLS}}$ in solving CEC'2006.

Algorithm		g01	g02	g03	g04	g05	g06	g07	g08	g09	g10	g11
CCiALF$_{\text{NLS}}$	Mn	**-15**	**-0.789524**	**-0.92708**	**-30665.5**	**5184.802**	**-6961.81**	**24.30625**	**-0.09583**	**680.6301**	**7049.272**	**0.749898**
	sd	1.23E-15	1.04E-02	1.99E-01	2.03E-05	1.13E+02	1.22E-02	1.70E-04	1.20E-17	1.16E-10	5.53E-02	2.05E-06
CCALF$_{\text{NLS}}$	Mn	-11.466814	-0.48845	inf	-28091.3	5842.945	-3387.15	30.81662	-0.06685	683.1372	19117.74	1.514531
	sd	7.78E-01	4.56E-02	inf	6.49E+02	3.93E+02	3.84E+02	1.45E+00	3.59E-02	6.55E-01	3.98E+03	1.08E+00
	h	1	1	-	1	1	1	1	1	1	1	1
Algorithm		g12	g13	g14	g15	g16	g17	g18	g19	g21	g23	g24
CCiALF$_{\text{NLS}}$	Mn	**-1**	**0.511189**	**-47.582**	962.6993	**-1.90516**	8939.882	**-0.82779**	**32.68655**	**240.2353**	**-282.256**	**-5.5034**
	sd	0.00E+00	2.64E-01	5.23E-01	1.56E+00	2.61E-08	5.58E+01	7.81E-02	1.03E-01	5.09E+01	1.07E+02	7.86E-08
CCALF$_{\text{NLS}}$	Mn	-0.96133	0.901668	inf	963.7127	-1.1045	8985.492	-0.33879	37.00475	545.5552	633.0714	-3.61184
	sd	3.09E-02	2.20E-01	inf	2.20E+00	1.83E-01	1.01E+02	1.40E-01	3.76E-01	2.15E+02	2.70E+02	2.38E-02
	h	1	1	-	0	1	0	1	1	1	1	1

5.2. Comparing CCiALF with other algorithms

Tables 4 and 5 present the results of CCiALF in comparison with seven state-of-the-art algorithms found in literature, on the CEC'2006 test functions set. In Tables 4 and 5, abbreviations are used for best solution (b), mean(Mn), worst solution (w), standard deviation (sd) and average number

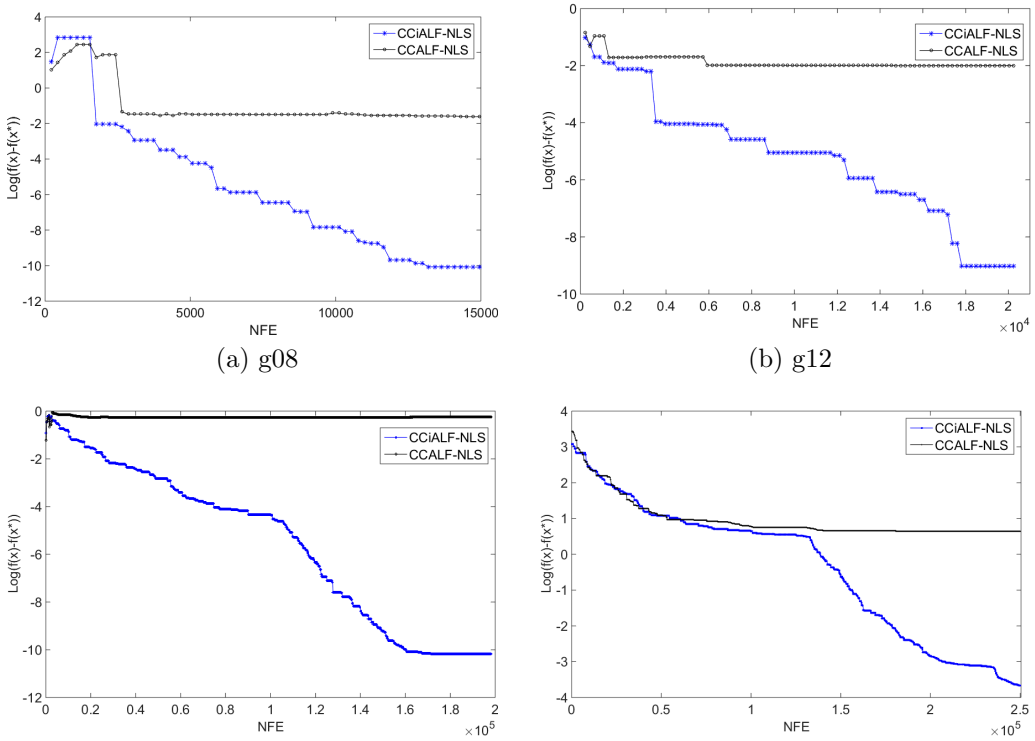


Figure 4: Convergence graph for $\text{CCiALF}_{\text{NLS}}$ and $\text{CCALF}_{\text{NLS}}$ algorithms

of function evaluations (NFE). In Tables 4 and 5, "N/A" indicates that the results are not reported in the cited algorithms and the best mean (Mn) values are presented in boldface. The performance of the proposed CCiALF is compared with P-Bf AFSP[36], COMDE [27], A-DDE[25], PSGA[14], APF-GA[45], M-ABC[24] and CB-ABC [5] algorithms. The results provided by these approaches were directly taken from cited authors' works.

The t-test with the significance level of 0.05 is performed according to Eq. (12). After performing the pairwise t-test for CCiALF and other algorithms, if CCiALF is equal, superior or inferior to the compared algorithm, then h is equal to 0, 1 and -1, respectively. For instance in Table 4, problem g01, column CB-ABC the value of h is equal to zero. Performing the pairwise t-test demonstrates CCiALF and CB-ABC performed similarly in problem g05.

For each column in Tables 4 and 5, the number of times h=1, h=0 and

Table 4: Comparison of results for test functions g01 to g13.

Prob/ Opt value		P-Bf AFSP[36]	COMDE [27]	A-DDE[25]	PSGA[14]	APF-GA[45]	M-ABC[24]	CB-ABC [5]	CCiALF
	b	-14.99999	-15	-15	-15	-15	-15	-15	-15
	Mn	-14.99992	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**
g01	w	N/A	-15	-15	-15	N/A	-15	-15	-14.99999
-15.0000	sd	2.30E-05	1.97E-13	7.00E-06	0	0.00E+00	0	5.03E-15	2.39E-08
	h	1	0	0	0	0	0	0	
	NFE	48,929	130,000	180,000	100,000	500,000	20,500	135,180	30,819
	b	-0.764816	-0.803619	-0.803605	-0.803597	-0.803601	-0.803615	-0.8036191	-0.8036176
	Mn	-0.730774	-0.801238	-0.77109	-0.794836	**-0.803518**	-0.799336	-0.7945223	-0.7930875
g02	w	N/A	-0.785265	-0.609853	-0.786442	N/A	-0.777438	-0.777844	-0.7617067
-0.803619	sd	1.80E-02	5.00E-03	3.66E-02	5.64E-03	1.00E-04	6.84E-03	8.32E-03	8.30E-03
	h	1	-1	1	0	-1	-1	0	
	NFE	104,312	200,000	180,000	100,000	500,000	83,500	198,270	240,000
	b	-1.000008	-1.000000	-1	-1.0005	-1.001	-1	-1.0005	-1.000501
	Mn	-0.999575	-1.000000	-1	-1.0005	**-1.001**	-1	-1.0005	-1.000501
g03	w	N/A	-0.9999999	-1	-1.0005	N/A	-1	-1.0005	-1.000500
-1.0005	sd	4.70E-04	3.03E-08	9.3E-12	2.69E-05	0.00E+00	4.68E-05	3.64E-07	1.69E-08
	h	1	1	1	0	-1	1	0	
	NFE	51,994	150,000	180,000	100,000	500,000	189,000	90,090	87,860
	b	-30665.538	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539
	Mn	-30665.524	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**
g04	w	N/A	-30665.539	-30665.539	-30665.539	N/A	-30665.539	-30665.539	-30665.539
-30665.5387	sd	1.00E-02	0	3.20E-13	7.28E-12	1.00E-04	2.22E-11	8.72E-11	9.80E-06
	h	1	0	0	0	0	0	0	
	NFE	102,188	50,000	180,000	100,000	500,000	76,400	45,045	26,268
	b	5126.498	5126.498	5126.497	5126.497	5126.497	5126.736	5126.497	5126.4967
	Mn	5128.478	5126.498	**5126.497**	5140.897	5127.542	5178.139	**5126.497**	**5126.497**
g05	w	N/A	5126.498	5126.497	5166.438	N/A	5317.196	5126.497	5126.497
5126.4967	sd	1.50E+00	0	2.10E-11	1.44E+01	1.43E+00	5.61E+01	1.07E-10	9.17E-08
	h	1	1	0	1	1	1	0	
	NFE	112,853	200,000	180,000	100,000	500,000	N/A	135180	156,248
	b	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814
	Mn	-6961.813	**-6961.814**	**-6961.814**	**-6961.814**	**-6961.814**	**-6961.814**	**-6961.814**	**-6961.814**
g06	w	N/A	-6961.81388	-6961.814	-6961.814	N/A	-6961.814	-6961.814	-6961.814
-6961.8139	sd	6.20E-04	0	2.11E-12	9.26E-12	0.00E+00	0	1.82E-12	5.19E-11
	h	1	0	0	0	0	0	0	
	NFE	106,718	12,000	180,000	100,000	500,000	107,000	45,000	17,573
	b	24.6325	24.3062	24.306	24.36	24.3062	24.315	24.3062	24.3062
	Mn	25.4384	24.3062	**24.306**	24.738	24.3062	24.415	24.3062	24.3062
g07	w	N/A	24.3062	24.306	24.999	N/A	24.854	24.3062	24.3062
24.3062	sd	3.60E-01	4.70E-07	4.02E-05	2.30E-01	0.00E+00	1.24E-01	4.16E-7	6.82E-07
	h	1	0	-1	1	0	1	0	
	NFE	117,449	200,000	180,000	100,000	500,000	N/A	135,180	8,745
	b	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.09582505
	Mn	-0.095824	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	**-0.09582505**
g08	w	N/A	-0.095825	-0.095825	-0.095825	N/A	-0.095825	-0.095825	-0.09582505
-0.095825	sd	3.40E-07	9.00E-18	9.10E-10	3.84E-09	0.00E+00	4.23E-17	2.87E-17	1.07E-15
	h	1	1	1	1	1	1	1	
	NFE	8,967	4,000	180,000	100,000	500,000	1,550	8,000	4,812
	b	680.6491	680.6300	680.63	680.631	680.63	680.632	680.63	680.6300
	Mn	680.6674	**680.6300**	**680.63**	680.658	**680.63**	680.647	**680.63**	**680.6300**
g09	w	N/A	680.6300	680.63	680.725	N/A	680.691	680.63	680.6300
680.630057	sd	8.60E-03	4.07E-13	1.15E-10	2.48E-02	0.00E+00	1.55E-02	2.77E-09	5.43E-08
	h	1	0	0	1	0	1	0	
	NFE	106,406	70,000	180,000	100,000	500,000	N/A	45,045	12,801
	b	7077.524	7049.248	7049.248	7049.255	7049.248	7051.706	7049.248	7049.248
	Mn	7198.382	**7049.248**	**7049.248**	7059.107	7077.682	7233.882	**7049.248**	**7049.248**
g10	w	N/A	7049.248615	7049.248	7092.609	N/A	7473.109	7049.248	7049.248
7049.248	sd	5.10E+01	1.50E-04	3.23E-04	12.6	5.12E+01	1.10E+02	3.98E-05	6.04E-07
	h	1	0	0	1	1	1	0	
	NFE	125,880	200,000	180,000	100,000	500,000	N/A	135,180	2,858
	b	0.7499	0.7499999	0.75	0.7499	0.7499	0.75	0.7499	0.7498959
	Mn	0.749901	0.7499999	0.75	0.7499	0.7499	0.75	0.7499	**0.7498984**
g11	w	N/A	0.7499999	0.75	0.7499	N/A	0.75	0.7499	0.7499000
0.749900	sd	8.10E-07	0	5.35E-15	1.92E-07	0.00E+00	2.3E-05	1.29E-10	2.04944E-16
	h	1	1	1	1	1	1	1	
	NFE	75,997	50,000	180,000	100,000	500,000	189,000	90,090	168,448
	b	-1	-1	-1	-1	-1	-1	-1	-1
	Mn	-0.999998	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**
g12	w	N/A	-1	-1	-1	N/A	-1	-1	-1
-1.000	sd	6.50E-07	0	4.10E-09	2.76E-09	0	0	0	0
	h	1	0	0	0	0	0	0	
	NFE	11,494	6,000	180,000	100,000	500,000	1,350	13,500	17,892
	b	0.056265	0.0539415	0.053942	0.054103	0.053942	0.053985	0.053942	0.05394151
	Mn	0.289244	**0.0539415**	0.079627	0.061601	0.053942	0.158552	0.06677	0.05394261
g13	w	N/A	0.0539415	0.438803	0.083042	N/A	0.442905	0.4388	0.05394986
0.0539415	sd	1.30E-01	1.40E-17	9.60E-02	7.23E-03	0.00E+00	1.73E-01	6.91E-02	4.03E-06
	h	1	0	0	1	0	1	0	
	NFE	95,508	150,000	180,000	100,000	500,000	189,000	198,270	19,883

h=-1 are counted and summarised in Table 6 as *Superior*, *Equal* and *Inferior*, respectively. For instance, in Table 6 the column PSGA, 16-6-0 means that CCiALF is superior 16 times, equal 6 times, and inferior 0 time, when compared with PSGA. Fig. 5 shows the computed scores of the CCiALF in comparison with other algorithms.

As shown in Fig. 5, CCiALF significantly outperforms P-Bf AFSP,

Table 5: Comparison of results for test functions g14 to g24 .

Prob/ Opt value		PSGA[14]	APF-GA[45]	M-ABC[24]	CB-ABC [5]	CCiALF
	b	-47.738	-47.7648	-47.641	-47.7649	-47.7649
	Mn	-47.679	-47.7648	-47.271	**-47.7649**	**-47.7649**
g14	w	-47.567	N/A	-46.537	-47.7648	-47.7649
-47.765	sd	3.74E-02	1.00E-04	2.46E-01	1.02E-05	4.04E-08
	h	1	1	1	0	
	NFE	100,000	500,000	N/A	239,715	152,697
	b	961.715	961.7150	961.715	961.715	961.7150
	Mn	961.732	**961.7150**	961.719	**961.715**	**961.7150**
g15	w	961.769	N/A	961.793	961.715	961.7150
961.715	sd	1.74E-02	0.00E+00	1.42E-02	2.81E-11	1.86E-08
	h	1	0	0	0	
	NFE	100,000	500,000	189,000	135,180	77,910
	b	-1.905	-1.905155	-1.905	-1.905	-1.905155
	Mn	-1.905	**-1.905155**	-1.905	-1.905	**-1.905155**
g16	w	-1.905	N/A	-1.905	-1.905	-1.905155
-1.905	sd	4.68E-15	0.00E+00	4.52E-16	7.90E-11	9.77E-09
	h	1	0	1	1	
	NFE	100,000	500,000	23,300	45,045	196,196
	b	8855.704	8853.539	8866.618	8853.534	8857.447
	Mn	8944.808	**8888.487**	8987.459	8902.870	8916.856
g17	w	9009.484	N/A	9165.219	8941.941	8956.457
8853.540	sd	2.75E+01	2.90E+01	9.57E+01	3.74E-01	3.64E+01
	h	1	-1	1	-1	
	NFE	100,000	500,000	189,000	239,715	240,000
	b	-0.866025	-0.866025	-0.866025	-0.866025	-0.8660255
	Mn	-0.856956	-0.865925	-0.7950187	-0.866025	**-0.8660255**
g18	w	-0.814956	N/A	-0.672216	-0.866025	-0.8660249
-0.866025	sd	1.21E-02	0.00E+00	9.39E-02	1.72E-08	3.58E-07
	h	1	1	1	1	
	NFE	100,000	500,000	70,600	135180	8,742
	b	32.796	32.65559	33.285	32.6556	32.65561
	Mn	33.752	**32.65559**	34.267	32.6556	32.66077
g19	w	34.789	N/A	35.746	32.6557	32.75902
32.656	sd	6.61E-01	0.00E+00	6.31E-01	1.88E-05	2.35E-04
	h	1	-1	1	-1	
	NFE	100,000	500,000	N/A	198,270	240,000
	b	193.78	196.6330	266.5	193.725	193.7243
	Mn	241.603	199.5158	306.609	**193.725**	193.7352
g21	w	293.762	N/A	329.96	193.725	193.7586
193.725	sd	4.07E+01	2.36E+00	1.98E+01	2.17E-06	1.20E-02
	h	1	1	1	-1	
	NFE	100,000	500,000	N/A	198,270	240,000
	b	-400.052	-399.7624	-159.739	-400.055	-400.0551
	Mn	-193.642	-394.7627	-35.272	**-400.055**	-400.0536
g23	w	-12.929	N/A	109.01	-400.055	-400.0364
-400.055	sd	1.23E+02	3.87E+00	8.28E+01	6.89E-05	5.00E-03
	h	1	1	1	0	
	NFE	100,000	500,000	N/A	239,715	240,000
	b	-5.508	-5.508013	-5.508	-5.508	-5.508013
	Mn	-5.508	**-5.508013**	-5.508	-5.508	**-5.508013**
g24	w	-5.508	N/A	-5.508	-5.508	-5.508013
-5.508	sd	1.33E-08	0.00E+00	2.71E-15	7.15E-15	1.30E-08
	h	1	0	1	1	
	NFE	100,000	500,000	6,800	27,000	6,450

Table 6: Pairwise comparison of CCiALF algorithm and other algorithms.

	P-Bf AFSP[36]	COMDE [27]	A-DDE[25]	PSGA[14]	APF-GA[45]	M-ABC[24]	CB-ABC [5]
Superior	13(**100%**)	4(**31%**)	4(**31%**)	16(**73%**)	8(**37%**)	16(**73%**)	5(**23%**)
Equal	0(0%)	8(61%)	8(61%)	6(27%)	10(45%)	5(23%)	14(63%)
Inferior	0(0%)	1(8%)	1(8%)	0(0%)	4(18%)	1(4%)	3(14%)
Total functions	13	13	13	22	22	22	22

COMDE, A-DDE, PSGA, APF-GA and M-ABC algorithms. But the proposed algorithm wins 5 times, draws 14 times and loses 3 times when compared to CB-ABC (according to Table 6), suggesting that both CCiALF

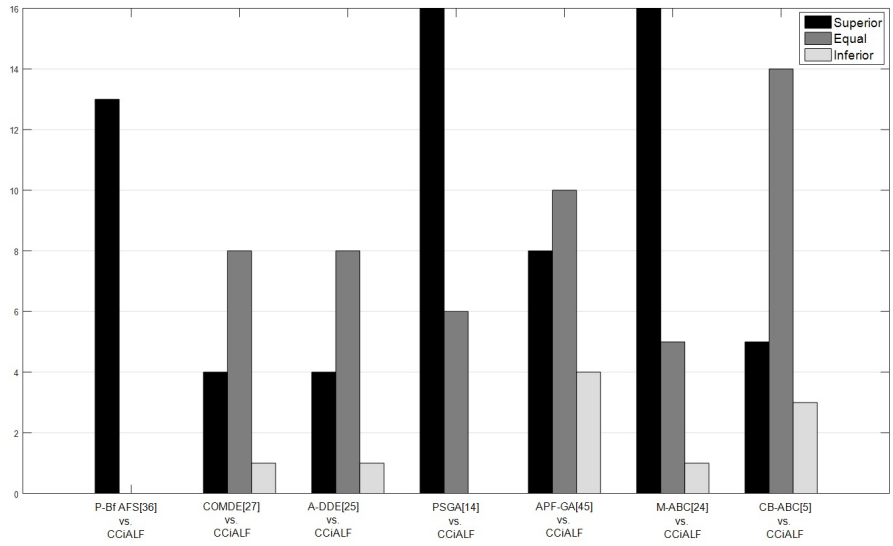


Figure 5: The number of "wins-draws-losses" of CCiALF compared with other algorithms.

and CB-ABC algorithms perform similarly. To compare both algorithms in terms of efficiency, their NFE are presented in Fig. 6. For 13 out of 22 test

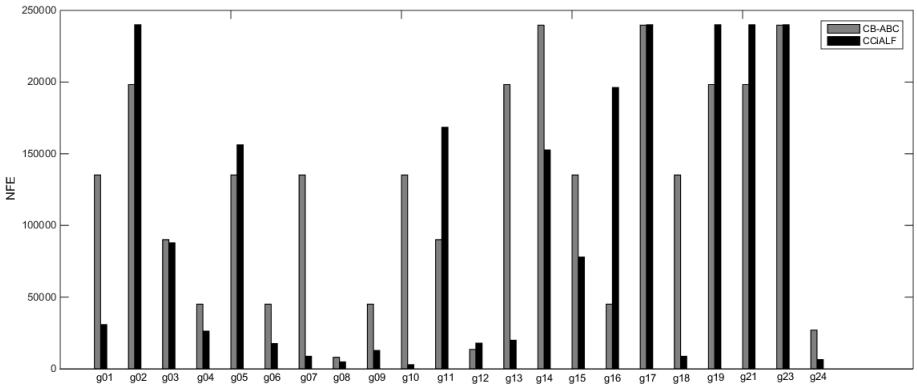


Figure 6: Comparison of CCiALF with CB-ABC [5] in terms of NFE.

functions (g01, g03, g04, g06, g07, g08, g09, g10, g13, g14, g15, g18 and g24), CCiALF has fewer NFE than CB-ABC. For these 13 test functions, CCiALF has superior or equal solution quality compared to CB-ABC. On the other hand, for the remaining problems (g02, g05, g11, g12, g16, g17, g19, g21 and g23) CCiALF has consumed more NFE than that of CB-ABC. In order to perform a fair comparison between CCiALF and CB-ABC, the

same MaxNFE that were used by CB-ABC[5] are used here to obtain the results shown in Table 7. In this case, CCiALF algorithm has similar or

Table 7: Comparison between CCiALF and CB-ABC with equal MaxNFE.

Algorithm		g02	g05	g11	g12	g16	g17	g19	g21	g23
CCiALF	Mn	-0.7929139	5126.497	0.7499	-1	-1.90514	8916.836	32.66290	193.7383	-400.0537
	sd	6.08E-03	6.17E-04	4.35E-07	3.64E-10	1.48E-07	2.27E+01	5.45E-02	2.45E-02	5.83E-05
CB-ABC	Mn	-0.7945223	5126.497	0.7499	-1	-1.905	8902.870	32.6556	193.725	-400.055
	sd	8.32E-03	1.07E-10	1.29E-10	0	7.90E-11	3.74E-01	1.88E-05	2.17E-06	6.89E-05
	h	0	0	0	0	1	-1	-1	-1	-1
MaxNFE		198,270	135,180	90,090	13,500	45,045	239,715	198,270	198,270	239,715

better performance than CB-ABC algorithm in g02, g05, g11, g12 and g16 while CCiALF lost the competition in g17, g19, g21 and g23. To sum up, according to Tables 4, 5 and 7, CCiALF has superior or equal solution quality compared to CB-ABC in 18 out of 22 test functions (g01, g02, g03, g04, g05, g06, g07, g08, g09, g10, g11, g12, g13, g14, g15, g16, g18 and g24) while using less or equal computation budget.

5.3. Results on engineering problems

In this section CCiALF algorithm is evaluated on five engineering problems from [34],[41],[4],[15] and [35]. The first problem is known as welded beam design [34]. In this problem the cost function is minimised subject to the stress constraint, physical constraints, buckling constraint, deflection constraint and side constraints. The problem is formulated as follows [8]:

$$
\begin{aligned}
\min \quad & f(\boldsymbol{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2) \\
\text{s.t.} \quad & g_1(\boldsymbol{x}) \equiv \tau_{max} - \tau(\boldsymbol{x}) \geq 0 \\
& g_2(\boldsymbol{x}) \equiv \sigma_{max} - \sigma(\boldsymbol{x}) \geq 0 \\
& g_3(\boldsymbol{x}) \equiv x_4 - x_1 \geq 0 \\
& g_4(\boldsymbol{x}) \equiv 5.0 - 0.10471 x_1^2 - 0.04811 x_3 x_4 (14.0 + x_2) \geq 0 \\
& g_5(\boldsymbol{x}) \equiv x_1 - 0.125 \geq 0 \\
& g_6(\boldsymbol{x}) \equiv \delta_{max} - \delta(\boldsymbol{x}) \geq 0 \\
& g_7(\boldsymbol{x}) \equiv P_c(\boldsymbol{x}) - 6000 \geq 0 \\
& 0.1 \leq (x_1, x_4) \leq 2, \;\; 0.1 \leq (x_2, x_3) \leq 10
\end{aligned}
\tag{13}
$$

where,

$$\tau(\boldsymbol{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}, \quad \tau' = \frac{P}{\sqrt{2}x_1x_2}, \quad \tau'' = \frac{MR}{J}$$

$$M = P\Big(L + \frac{x_2}{2}\Big), \quad R = \sqrt{\frac{x_2^2}{4} + \Big(\frac{x_1 + x_3}{2}\Big)^2}$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \Big(\frac{x_1 + x_3}{2}\Big)^2\right]\right\}, \quad \sigma(\boldsymbol{x}) = \frac{6PL}{x_3^2x_4}$$

$$\delta(\boldsymbol{x}) = \frac{4PL^3}{Ex_3^3x_4}, \quad P_c(\boldsymbol{x}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$$

where P=6000 lb, L=14 in., $\delta_{max} = 0.25$ in., $E = 30 \times 10^6$ psi, $G = 12 \times 10^6$ psi, $\tau_{max} = 13600$ psi and $\sigma_{max} = 30000$ psi. The results of CCiALF are compared with Coe02 [8], COMDE [27], MVDE [11] and CB-ABC [5] algorithms in Table 8. As shown in Table 8, CCiALF outperforms Coe02 and MVDE algorithms in terms of solution quality. Although CCiALF has similar solution quality as COMDE and CB-ABC algorithms, the NFE value for CCiALF is significantly fewer than other algorithms. The best objective function value found by CCiALF algorithm is $f_{CCiALF} = 1.724852$ with the decision vector:
$\boldsymbol{x}_{CCiALF} = (0.205729639785972, 3.470488665631285, 9.036623910355711, 0.205729639786304)$.
The reported decision variable is feasible and constraints are : $g_1(\boldsymbol{x}) = 7.24E-10$, $g_2(\boldsymbol{x}) = 2.00E-08$, $g_3(\boldsymbol{x}) = 3.32E-13$, $g_4(\boldsymbol{x}) = 3.432983785$, $g_5(\boldsymbol{x}) = 0.08072964$, $g_6(\boldsymbol{x}) = 0.235540323$ and $g_7(\boldsymbol{x}) = 1.88E-08$.

The second engineering problem is proposed by Sandgren [41] called pressure vessel design which tries to minimise the cost function for materials of forming and welding of pressure vessel. The four decision variables are x_1 (thickness of the shell), x_2 (thickness of the head), x_3 (inner radius) and x_4 (length of the cylindrical segment of the vessel). x_1 and x_2 can only be the integer multiples of 0.0625 inch [2]. Table 8 presents the statistical results for CCiALF, Coe02 [8], HCP [29], COMDE [27], MVDE [11] and CB-ABC [5] algorithms. The best found solution obtained by CCiALF is $f_{CCiALF} = 6059.7143350$ with the decision vector:
$\boldsymbol{x}_{CCiALF} = (0.8125, 0.4375, 42.098445595854810, 176.6365958424410)$
and the constraints are $g_1(\boldsymbol{x}) = 0$, $g_2(\boldsymbol{x}) = 3.59E-02$, $g_3(\boldsymbol{x}) = 0$ and $g_4(\boldsymbol{x}) = 63.36$ which demonstrates the obtained solution is feasible. The

Table 8: Results for engineering problems (Note that "N/A" denotes the results are not available and the best mean vales are presented in boldface).

Welded beam design						
Algorithm	best	mean	worst	std	NFE	h
Coe02[8]	1.728226	1.792654	1.993408	7.47E-02	80,000	1
COMDE [27]	1.724852	**1.724852**	1.724852	1.60E-12	20,000	0
MVDE [11]	1.7248527	1.7248621	1.7249215	7.88E-06	15,000	1
CB-ABC [5]	1.724852	**1.724852**	N/A	2.38E-11	15,000	0
CCiALF	1.724852	**1.724852**	1.724854	5.11E-07	10,000	
Pressure vessel design						
Algorithm	best	mean	worst	std	NFE	h
Coe02[8]	6059.946341	6177.253268	6469.32201	1.31E+02	80,000	1
HCP[29]	6059.69	6171.13	N/A	1.13E+02	30,000	1
COMDE [27]	6059.714335	**6059.714335**	6059.714335	3.62E-10	30,000	0
MVDE [11]	6059.714387	6059.997236	6090.533528	2.91E+00	15,000	0
CB-ABC [5]	6059.714335	6126.623676	N/A	1.14E+02	15,000	1
CCiALF	6059.714335	**6059.714335**	6059.714335	1.01E-11	12,000	
Weight of a tension/compression string						
Algorithm	best	mean	worst	std	NFE	h
Coe02[8]	0.012679	0.012742	0.012973	5.90E-05	80,000	1
HCP[29]	0.012679	0.0127	N/A	1.90E-05	650	1
COMDE [27]	0.012665233	0.012667168	0.012676809	3.09E-06	20,000	1
MVDE [11]	0.01266527172	0.012667324	0.012719055	2.45E-06	10,000	1
CB-ABC [5]	0.012665	0.012671	N/A	1.42E-05	15,000	1
CCiALF	0.01266523279	**0.012665251**	0.012665233	9.87E-08	5,000	
Speed reducer design						
Algorithm	best	mean	worst	std	NFE	h
COMDE [27]	2994.4710661	2994.4710661	2994.4710661	1.54E-12	21,000	0
MVDE [11]	2994.471066	**2994.471066**	2994.471069	2.82E-07	30,000	0
CB-ABC [5]	2994.471066	**2994.471066**	N/A	2.48E-07	15,000	0
CCiALF	2994.471066	**2994.471066**	2994.4710661	2.31E-12	10,000	
Three-bar truss design						
Algorithm	best	mean	worst	std	NFE	h
COMDE [27]	263.8958434	263.8958434	263.8958434	5.43E-13	21,000	0
MVDE [11]	263.89584337	263.89584338	263.8958548	2.58E-07	7,000	0
CCiALF	263.89584337	**263.89584337**	263.89584337	4.23E-14	5000	

CCiALF algorithm outperforms Coe02, HCP and CB-ABC algorithms in terms of solution quality and is more efficient than COMDE and MVDE algorithms. This problem is formulated as follows [29]:

$$
\begin{aligned}
\min \quad & f(\boldsymbol{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \\
\text{s.t.} \quad & g_1(\boldsymbol{x}) = -0.0193x_3 + x_1 \geq 0 \\
& g_2(\boldsymbol{x}) = -0.00954x_3 + x_2 \geq 0 \\
& g_3(\boldsymbol{x}) = -1296000 + \pi x_3^2 x_4 + \frac{4}{3}\pi x_3^3 \geq 0 \\
& g_4(\boldsymbol{x}) = -x_4 + 240 \geq 0 \\
& 0.0625 \leq x_1 \leq 6.1875, \;\; 0.0625 \leq x_2 \leq 6.1875, \\
& 10 \leq x_3 \leq 200, \;\; 10 \leq x_4 \leq 200
\end{aligned}
\tag{14}
$$

The third engineering problem is minimisation of weight of a tension/ compression sting presented by [4] subject to constraints on minimum deflection, shear stress, surge frequency and limits on outside diameter. The formulation of the problem is as follows [29]:

$$
\begin{aligned}
\min \quad & f(\boldsymbol{x}) = (x_3 + 2)x_2x_1^2 \\
\text{s.t.} \quad & g_1(\boldsymbol{x}) = \frac{x_2^3 x_3}{71785x_1^4} - 1 \geq 0 \\
& g_2(\boldsymbol{x}) = 1 - \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} - \frac{1}{5108x_1^2} \geq 0 \\
& g_3(\boldsymbol{x}) = \frac{140.45x_1}{x_2^2x_3} - 1 \geq 0 \\
& g_4(\boldsymbol{x}) = 1 - \frac{x_1 + x_2}{1.5} \geq 0 \\
& 0.05 \leq x_1 \leq 2, \;\; 0.25 \leq x_2 \leq 1.3, \;\; 2 \leq x_3 \leq 15
\end{aligned}
\tag{15}
$$

The results for CCiALF, Coe02 [8], HCP [29] COMDE [27], MVDE [11] and CB-ABC [5] algorithms are presented in Table 8. As shown in Table 8, CCiALF slightly improves the best known objective function value and the proposed algorithm outperforms other algorithms. The obtained objective function value is $f_{CCiALF} = 0.01266523279$ and the corresponding decision vector and constraints are as follows:
$\boldsymbol{x} = (0.051689059696547, 0.356717706450724, 11.288967706745048)$, $g_1(\boldsymbol{x}) = 2.22E-16$, $g_2(\boldsymbol{x}) = 1.11E-16$, $g_3(\boldsymbol{x}) = 4.05$ and $g_4(\boldsymbol{x}) = 7.28E-01$.

The fourth engineering problem is introduced by Golinski [15] and known as speed reducer design. The problem involves the minimisation of the weight

of speed reducer. The speed reducer problem has seven decision variables and eleven constraints. The formula of speed reducer design is as follows:

$$
\begin{aligned}
\min \quad & f(\boldsymbol{x}) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1 \\
& \qquad (x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2) \\
\text{s.t.} \quad & g_1(\boldsymbol{x}) = 1 - \frac{27}{x_1x_2^2x_3} \geq 0 \\
& g_2(\boldsymbol{x}) = 1 - \frac{397.5}{x_1x_2^2x_3^2} \geq 0 \\
& g_3(\boldsymbol{x}) = 1 - \frac{1.93x_4^3}{x_2x_3x_6^4} \geq 0 \\
& g_4(\boldsymbol{x}) = 1 - \frac{1.93x_5^3}{x_2x_3x_7^4} \geq 0 \\
& g_5(\boldsymbol{x}) = 1 - \frac{1}{110x_6^3}\sqrt{\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6} \geq 0 \\
& g_6(\boldsymbol{x}) = 1 - \frac{1}{85x_7^3}\sqrt{\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5 \times 10^6} \geq 0 \\
& g_7(\boldsymbol{x}) = 1 - \frac{x_2x_3}{40} \geq 0 \\
& g_8(\boldsymbol{x}) = 1 - \frac{5x_2}{x_1} \geq 0 \\
& g_9(\boldsymbol{x}) = 1 - \frac{x_1}{12x_2} \geq 0 \\
& g_{10}(\boldsymbol{x}) = 1 - \frac{(1.5x_6 + 1.9)}{x_4} \geq 0 \\
& g_{11}(\boldsymbol{x}) = 1 - \frac{(1.1x_7 + 1.9)}{x_5} \geq 0 \\
& 2.6 \leq x_1 \leq 3.6, \;\; 0.7 \leq x_2 \leq 0.8, \;\; 17 \leq x_3 \leq 28, \;\; 7.3 \leq x_4 \leq 8.3 \\
& 7.8 \leq x_5 \leq 8.3, \;\; 2.9 \leq x_6 \leq 3.9, \;\; 5 \leq x_7 \leq 5.5
\end{aligned}
\tag{16}
$$

The problem is solved by CCiALF and the results are compared with COMDE [27], MVDE [11] and CB-ABC [5] algorithms in Table 8. Although the performance of CCiALF algorithm is similar to other algorithms, the NFE value is significantly fewer than others. The decision variable vector and the constraints are as follows:

$\boldsymbol{x} = (3.505071907646737, 0.700070905298447, 17.000572946016090, 7.413638851442300, 7.824441935611341, 3.352677467256310, 5.287622983296388)$, $g_1(\boldsymbol{x}) = 7.39E-02$, $g_2(\boldsymbol{x}) = 1.98E-01$, $g_3(\boldsymbol{x}) = 4.92E-01$, $g_4(\boldsymbol{x}) = 9.4E-01$, $g_5(\boldsymbol{x}) = 7.79E-12$, $g_6(\boldsymbol{x}) = 5.65E-12$, $g_7(\boldsymbol{x}) = 7.02E-01$, $g_8(\boldsymbol{x}) = 3.27E-12$, $g_9(\boldsymbol{x}) = 5.83E-01$, $g_{10}(\boldsymbol{x}) = 5.13E-02$ and $g_{11}(\boldsymbol{x}) = 1.08E-11$ with $f_{CCiALF} = 2994.471066$.

The last engineering problem is called three-bar truss design [35]. The optimisation problem consists of minimising the volume of a loaded 3-bar truss, subject to stress constraints on each of the truss members. Eq. (17) presents the formulation of the three-bar truss design.

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) = (2\sqrt{2}x_1 + x_2) \times 100 \\ \text{s.t.} \quad & g_1(\boldsymbol{x}) = 2 - 2 \times \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} \geq 0 \\ & g_2(\boldsymbol{x}) = 2 - 2 \times \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} \geq 0 \\ & g_3(\boldsymbol{x}) = 2 - 2 \times \frac{1}{x_1 + \sqrt{2}x_2} \geq 0 \\ & 0 \leq x_1 \leq 1, \ \ 0 \leq x_2 \leq 1 \end{aligned} \tag{17}$$

As shown in Table 8, CCiALF algorithm has the same performance as COMDE [27] and MVDE [11] algorithms with fewer number of function evaluations. The decision vector and the constraints obtained by CCiALF are as follows: $\boldsymbol{x} = (0.788675131095601, 0.408248300361130)$, $g_1(\boldsymbol{x}) = 0$, $g_2(\boldsymbol{x}) = 1.46E{+}00$ and $g_3(\boldsymbol{x}) = 5.36E-01$ with $f_{CCiALF} = 263.89584337$.

To sum up, CCiALF algorithm slightly improves the objective function value for the third engineering problem, whereas the CCiALF reaches the best known objective value in other engineering problems. In all engineering problems CCiALF uses remarkably fewer NFE in comparison to other algorithms. Particularly, in the four engineering problems which CCiALF and CB-ABC are compared, the proposed CCiALF algorithm is equal or superior to CB-ABC algorithm in terms of solution quality with less computational cost. In short, CCiALF demonstrates promising results in obtaining high quality solutions for these engineering problems while the NFE for solving all engineering problems are fewer than other algorithms.

6. Conclusion

In this paper a CCiALF algorithm has been proposed to solve constrained optimisation problems. The proposed algorithm employs the cooperative coevolutionary framework that coevolves two populations, one for the decision variables, and the other for Lagrangian multipliers. We have demonstrated that the conventional ALF method has an issue of distorting the objective values for solutions in feasible regions and to remedy this problem, we have proposed an improved ALF method (iALF), which can reduce the amount of distortion caused in the ALF. In order to demonstrate the capability of the proposed iALF, $\text{CCiALF}_{\text{NLS}}$ and $\text{CCALF}_{\text{NLS}}$ algorithms are applied to solve the CEC'2006 test functions. The experimental results have shown that $\text{CCiALF}_{\text{NLS}}$ significantly outperforms $\text{CCALF}_{\text{NLS}}$ in terms of constraint handling efficiency, solution quality and convergence speed. Following that, this paper also shows that CCiALF algorithm is equal or superior to the state-of-the-art algorithms in terms of solution quality and superior in efficiency. Another substantial advantage of CCiALF algorithm is its ability to deal with black-box optimisation problems as well as solving real-world engineering problems in an efficient way. Our future work will be looking into applying CCiALF algorithm to solve more challenging real-world constrained optimisation problems.

Acknowledgement

We are most grateful to all reviewers and the associate editor for their constructive comments which have helped to improve this paper substantially. Melih Ozlen is supported by the Australian Research Council under the Discovery Projects funding scheme (project DP140104246).

References

[1] Adeli, H., Cheng, N.-T., 1994. Augmented lagrangian genetic algorithm for structural optimization. Journal of Aerospace Engineering 7 (1), 104–118.

[2] Akay, B., Karaboga, D., 2012. Artificial bee colony algorithm for large-scale problems and engineering design optimization. Journal of Intelligent Manufacturing 23 (4), 1001–1014.

[3] Barbosa, H. J., 1999. A coevolutionary genetic algorithm for constrained optimization. In: Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on. Vol. 3. IEEE.

[4] Belegund, A., 1982. A study of mathematical programming methods for structural optimization. PhD Thesis,Department of Civil and Environmental Engineering, University of Iowa, Iowa.

[5] Brajevic, I., 2015. Crossover-based artificial bee colony algorithm for constrained optimization problems. Neural Computing and Applications 26 (7), 1587–1601.

[6] Brest, J., Greiner, S., Bošković, B., Mernik, M., Zumer, V., 2006. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. Evolutionary Computation, IEEE Transactions on 10 (6), 646–657.

[7] Cheng, J., Zhang, G., Caraffini, F., Neri, F., 2015. Multicriteria adaptive differential evolution for global numerical optimization. Integrated Computer-Aided Engineering 22 (2), 103–107.

[8] Coello, C. A. C., Montes, E. M., 2002. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Advanced Engineering Informatics 16 (3), 193–203.

[9] Costa, M. F. P., Rocha, A. M. A., Fernandes, E. M., 2014. An artificial fish swarm algorithm based hyperbolic augmented lagrangian method. Journal of Computational and Applied Mathematics 259, 868–876.

[10] Courant, R., et al., 1943. Variational methods for the solution of problems of equilibrium and vibrations. Bull. Amer. Math. Soc 49 (1), 1–23.

[11] De Melo, V. V., Carosio, G. L., 2013. Investigating multi-view differential evolution for solving constrained engineering design problems. Expert Systems with Applications 40 (9), 3370–3377.

[12] Deb, K., 2000. An efficient constraint handling method for genetic algorithms. Computer methods in applied mechanics and engineering 186 (2), 311–338.

[13] Deb, K., Srivastava, S., 2012. A genetic algorithm based augmented lagrangian method for constrained optimization. Computational Optimization and Applications 53 (3), 869–902.

[14] Dhadwal, M. K., Jung, S. N., Kim, C. J., 2014. Advanced particle swarm assisted genetic algorithm for constrained optimization problems. Computational Optimization and Applications 58 (3), 781–806.

[15] Golinski, J., 1974. An adaptive optimization system applied to machine synthesis. Mechanism and Machine Theory 8 (4), 419–436.

[16] Hestenes, M. R., 1969. Multiplier and gradient methods. Journal of optimization theory and applications 4 (5), 303–320.

[17] Iacca, G., Caraffini, F., Neri, F., 2015. Continuous parameter pools in ensemble differential evolution. In: Computational Intelligence, 2015 IEEE Symposium Series on. IEEE, pp. 1529–1536.

[18] Iacca, G., Neri, F., Caraffini, F., Suganthan, P. N., 2014. A differential evolution framework with ensemble of parameters and strategies and pool of local search algorithms. In: Applications of Evolutionary Computation. Springer, pp. 615–626.

[19] Krohling, R., dos Santos Coelho, L., et al., 2006. Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 36 (6), 1407–1416.

[20] Liang, J., Runarsson, T. P., Mezura-Montes, E., Clerc, M., Suganthan, P., Coello, C. C., Deb, K., 2006. Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. Journal of Applied Mechanics 41, 8.

[21] Mahdavi, A., Shiri, M. E., 2015. An augmented lagrangian ant colony based method for constrained optimization. Computational Optimization and Applications 60 (1), 263–276.

[22] Mallipeddi, R., Suganthan, P. N., 2010. Ensemble of constraint handling techniques. Evolutionary Computation, IEEE Transactions on 14 (4), 561–579.

[23] mathworks.com.au, 2015. Find minimum of constrained nonlinear multivariable function - matlab fmincon. [Online]. Available: http://au.mathworks.com/help/optim/ug/fmincon.html.

[24] Mezura-Montes, E., Cetina-Domínguez, O., 2012. Empirical analysis of a modified artificial bee colony for constrained numerical optimization. Applied Mathematics and Computation 218 (22), 10943–10973.

[25] Mezura-Montes, E., Palomeque-Ortiz, A. G., 2009. Self-adaptive and deterministic parameter control in differential evolution for constrained optimization. In: Constraint-Handling in Evolutionary Optimization. Springer, pp. 95–120.

[26] Michalewicz, Z., Schoenauer, M., 1996. Evolutionary algorithms for constrained parameter optimization problems. Evolutionary computation 4 (1), 1–32.

[27] Mohamed, A. W., Sabry, H. Z., 2012. Constrained optimization based on modified differential evolution algorithm. Information Sciences 194, 171–208.

[28] Nash, S. G., Sofer, A., 1996. Linear and nonlinear programming. McGraw-Hill series in industrial engineering and management science. McGraw-Hill, New York.

[29] Nema, S., Goulermas, J. Y., Sparrow, G., Helman, P., 2011. A hybrid cooperative search algorithm for constrained optimization. Structural and Multidisciplinary Optimization 43 (1), 107–119.

[30] Potter, M. A., De Jong, K. A., 1994. A cooperative coevolutionary approach to function optimization. In: Parallel problem solving from nature—PPSN III. Springer, pp. 249–257.

[31] Powell, M. J., 1969. A method for non-linear constraints in minimization problems. in: R. Fletcher, Ed., Optimization, Academic Press, London/New York, pp. 283–298.

[32] Qin, A. K., Huang, V. L., Suganthan, P. N., 2009. Differential evolution algorithm with strategy adaptation for global numerical optimization. Evolutionary Computation, IEEE Transactions on 13 (2), 398–417.

[33] Qin, A. K., Suganthan, P. N., 2005. Self-adaptive differential evolution algorithm for numerical optimization. In: Evolutionary Computation, 2005. The 2005 IEEE Congress on. Vol. 2. IEEE, pp. 1785–1791.

[34] Ragsdell, K., Phillips, D., 1976. Optimal design of a class of welded structures using geometric programming. Journal of Manufacturing Science and Engineering 98 (3), 1021–1025.

[35] Ray, T., Liew, K. M., 2003. Society and civilization: An optimization algorithm based on the simulation of social behavior. Evolutionary Computation, IEEE Transactions on 7 (4), 386–396.

[36] Rocha, A. M. A., Costa, M. F. P., Fernandes, E. M., 2014. A filter-based artificial fish swarm algorithm for constrained global optimization: theoretical and practical issues. Journal of Global Optimization 60 (2), 239–263.

[37] Rocha, A. M. A., Martins, T. F., Fernandes, E. M., 2011. An augmented lagrangian fish swarm based method for global optimization. Journal of computational and applied mathematics 235 (16), 4611–4620.

[38] Rockafellar, R. T., 1973. A dual approach to solving nonlinear programming problems by unconstrained optimization. Mathematical Programming 5 (1), 354–373.

[39] Rockafellar, R. T., 1973. The multiplier method of hestenes and powell applied to convex programming. Journal of Optimization Theory and applications 12 (6), 555–562.

[40] Salcedo-Sanz, S., 2009. A survey of repair methods used as constraint handling techniques in evolutionary algorithms. Computer science review 3 (3), 175–192.

[41] Sandgren, E., 1990. Nonlinear integer and discrete programming in mechanical design optimization. Journal of Mechanical Design 112 (2), 223–229.

[42] Sarma, K. C., Adeli, H., 2000. Fuzzy genetic algorithm for optimization of steel structures. Journal of Structural Engineering 126 (5), 596–604.

[43] Storn, R., Price, K., 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization 11 (4), 341–359.

[44] Tahk, M.-J., Sun, B.-C., 2000. Coevolutionary augmented lagrangian methods for constrained optimization. Evolutionary Computation, IEEE Transactions on 4 (2), 114–124.

[45] Tessema, B., Yen, G. G., 2009. An adaptive penalty formulation for constrained evolutionary optimization. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 39 (3), 565–578.

[46] Zhang, J., Sanderson, A. C., 2009. Jade: adaptive differential evolution with optional external archive. Evolutionary Computation, IEEE Transactions on 13 (5), 945–958.