Thank you for downloading this document from the RMIT Research Repository.

The RMIT Research Repository is an open access database showcasing the research outputs of RMIT University researchers.


**Citation:**


See this record in the RMIT Research Repository at:


**Version:** Accepted Manuscript

**Copyright Statement:** © 2016 Elsevier Inc.

**Link to Published Version:**

[http://dx.doi.org/10.1016/j.ins.2016.06.047](http://dx.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-
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:

\[
\min_{\mathbf{x}} f(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^p
\]  
\[
g_i(\mathbf{x}) \geq 0, \quad i = 1, ..., m
\]  
\[
h_j(\mathbf{x}) = 0, \quad j = 1, ..., n
\]  
\[
lb_k \leq x_k \leq ub_k, \quad k = 1, ..., p
\]
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 $x$. 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 $\theta$ function is presented as follows [13]:

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

(5)

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

(6)

where $R$ is a positive penalty parameter, $\mu$ is a $1 \times m$ multiplier, $\tau$ is a $1 \times n$ multiplier for inequality and equality constraints respectively, and $G$ can be any function or value. The $\theta$ 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, i.e., 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:

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

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


Figure 1: Four different regions divided by ALF.

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

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

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

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 \( w, x, y, z \) 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 \( \mu \) 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:

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

(8)

In this problem two points (\(x_A\) and \(x_B\)) 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, \(x_A\) and \(x_B\) are both feasible solutions satisfying all constraints and since \(f(x_A) \geq f(x_B)\), \(x_B\) is better than \(x_A\). Although \(x_B\) is better than \(x_A\), the ALF value for \(x_A\) and \(x_B\) shows that \(F(x_A) \leq F(x_B)\) incorrectly suggesting that \(x_A\) is better than \(x_B\).

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:

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

(9)
where if point $\mathbf{x}$ 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 $\mathbf{x}$ is an infeasible solution (located in region 4) $iF$ is set to equal to $f + R \sum (g_j(\mathbf{x})^2 + 2g_j(\mathbf{x})\mu_j)$. In other words, if $\mathbf{x}$ 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 $\mathbf{x}$ is in the infeasible area, the $R \sum (g_j(\mathbf{x})^2 + 2g_j(\mathbf{x})\mu_j)$ 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.

$$
\begin{array}{|l|c|c|c|c|}
\hline
\text{Points} & \mathbf{g(\mathbf{x})} & \mathbf{\mu} & \mathbf{\theta(\mathbf{g + \mu})} & \mathbf{F} & \mathbf{iF} \\
\hline
\mathbf{w} \quad \text{(Region 1)} & > 0 & < 0 & 0 & f(\mathbf{w}) - R\mu^2 & f(\mathbf{w}) \\
\hline
\mathbf{x} \quad \text{(Region 2)} & > 0 & < 0 & \mathbf{g(\mathbf{x}) + \mu} & f(\mathbf{x}) + R(\mathbf{g(\mathbf{x})}^2 + 2\mathbf{g(\mathbf{x})}\mu) & f(\mathbf{x}) \\
\hline
\mathbf{y} \quad \text{(Region 3)} & = 0 & < 0 & \mathbf{\mu} & f(\mathbf{y}) & f(\mathbf{y}) \\
\hline
\mathbf{z} \quad \text{(Region 4)} & < 0 & < 0 & \mathbf{\mu} & f(\mathbf{z}) + R(\mathbf{g(\mathbf{z})}^2 + 2\mathbf{g(\mathbf{z})}\mu) & f + R(\mathbf{g(\mathbf{z})}^2 + 2\mathbf{g(\mathbf{z})}\mu) \\
\hline
\end{array}
$$

The iALF is applied to calculate the fitness function for problem g24 in Table 2. The last row presents the $iF$ value for points $\mathbf{x}_A$ and $\mathbf{x}_B$. 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 $iF(\mathbf{x}_A) \geq iF(\mathbf{x}_B)$ which is the same as $f(\mathbf{x}_A) \geq f(\mathbf{x}_B)$, iALF will give us the correct result, i.e., $\mathbf{x}_B$ is better than $\mathbf{x}_A$.

$$
\begin{array}{|l|l|}
\hline
\text{Point } \mathbf{x}_A & \text{Point } \mathbf{x}_B \\
\hline
x_1 & 2.404678 & x_1 & 2.329524 \\
x_2 & 1.797169 & x_2 & 3.178475 \\
g_1(\mathbf{x}_A) & 2.096757 \geq 0 & g_1(\mathbf{x}_B) & 4.91E - 05 \geq 0 \\
g_2(\mathbf{x}_A) & 0.999992 \geq 0 & g_2(\mathbf{x}_B) & 2.03E - 07 \geq 0 \\
f(\mathbf{x}_A) & -4.201847 & f(\mathbf{x}_B) & -5.507999 \\
\mu_1 & -1 & \mu_1 & -1 \\
\mu_2 & -1 & \mu_2 & -1 \\
R & 50000 & R & 50000 \\
F(\mathbf{x}_A) & -100004.2018 & F(\mathbf{x}_B) & -10.440193 \\
iF(\mathbf{x}_A) & -4.201847 & iF(\mathbf{x}_B) & -5.508012 \\
\hline
\end{array}
$$

Fig. 2 demonstrates the distortion of the fitness function in neighbourhood of points $\mathbf{x}_A$ and $\mathbf{x}_B$ by ALF. As shown in Fig. 2, the $iF$ and $f$ values
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 a 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 ($\mathbf{x}$ and $\mathbf{\mu}$). The first population (PopulationI) evolves the decision vector $\mathbf{x}$, whereas the second population (PopulationII) evolves the multiplier vector $\mathbf{\mu}$. 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., $\mathbf{x}_{i,It} = (x_{1,i,It}, \ldots, x_{p,i,It})$, $i = 1, \ldots, N_{pop}$ from one generation to the next. The initial population can be generated as follows:

$$\mathbf{x}_{i,1} = \mathbf{lb} + \text{rand}(0,1)(\mathbf{ub} - \mathbf{lb}); i = 1, 2, \cdots, N_{pop}$$  \hspace{1cm} (10)$$

where $\text{rand}(0,1)$ is a uniform random variable in [0,1] and $\mathbf{ub}, \mathbf{lb}$ are upper and lower bound vectors for $\mathbf{x}$, respectively.
After initialisation phase, a mutation operator is performed on the target vector $\mathbf{x}_{i,t}$ to generate a mutant vector $\mathbf{v}_{i,t}$. There are different strategies to produce a mutant vector [32]. After mutant vectors are generated, the crossover operator should be performed on $\mathbf{x}_{i,t}$ and its corresponding mutant vector $\mathbf{v}_{i,t}$ to produce a trial vector ($\mathbf{u}_{i,t}$). 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 $[\mathbf{lb}, \mathbf{ub}]$. Then the fitness function of the trial vector $\mathbf{u}_{i,t}$ and the corresponding target vector $\mathbf{x}_{i,t}$ should be compared and the next generation should be formed. The binary selection procedure is as follows:

$$\mathbf{x}_{i,t+1} = \begin{cases} \mathbf{u}_{i,t}, & \text{iff } (\mathbf{u}_{i,t}) \leq f(\mathbf{x}_{i,t}) \\ \mathbf{x}_{i,t}, & \text{otherwise} \end{cases}$$

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: $\mathbf{x}$, the decision variable vector for the problem to be optimised, and $\mathbf{\mu}$, the Lagrangian multipliers. PopulationI
and PopulationII evolve $\mathbf{x}$ and $\boldsymbol{\mu}$, respectively. In Fig. 3, $m$, $n$, $p$, $N_{\text{pop}1}$, $N_{\text{pop}2}$ 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 ($\mathbf{x}$) and PopulationII ($\boldsymbol{\mu}$) should be initialised randomly according to Eq. (10). Note that for populationII, $lb$, $ub$ are $\boldsymbol{\mu}_0$ and 0, respectively. In order to evaluate the fitness function for PopulationI and II, members of both populations should interact and $\mathbf{x}$, $\boldsymbol{\mu}$ 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 ($\mathbf{x}_{i,It}; i = 1, \ldots, N_{\text{pop}1}$) 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 ($\boldsymbol{\mu}_{j,It}; j = 1, \ldots, N_{\text{pop}2}$) is combined with the best individual in PopulationI ($\mathbf{x}^*$). 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 $\mathbf{x}^*$ 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 $[lb, ub]$. Then by combining two subpopulations, the fitness function value
is calculated and the best individual from each sub-population \((x^*, \mu^*)\) 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 \(\beta_2\) iterations, \(R\) is updated according to the method presented in [13].

In addition, to improve the convergence speed, in every \(\beta_1\) iterations, if the difference between the best objective value in two successive iterations is less than \(\delta\), 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 \(\beta_3\) successive iterations. The algorithm stops when it reaches MaxNFE or it has not improved for \(\beta_3\) 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

1: procedure CCiALF($N_{pop1}, N_{pop2}, It_{max}, \mu_0, \beta_1, \beta_2, \delta, Const, MaxNFE$)
2:     \textbf{Initialisation}
3:     Randomly generate $Population_{I,0}$ and $Population_{II,0}$ (Eq. (10))
4:     Initialise $R_0$ as follows : $R_0 = \frac{\sum_{i=1}^{N_{pop1}} |f_i(x)|}{\sum_{i=1}^{N_{pop1}} \sum_{j=1}^{n+m+\phi(g}_j(x))}$; $R_{old} = R_0$
5:     Calculate $iF_{1,0}(x, \mu), iF_{2,0}(x^*, \mu)$ for PopulationI,II respectively (Eq. (9))
6:     Add $NFE_0$
7:     $BFS(1) = x^*$
8:     Select $(x^*, \mu^*)$
9:     \textbf{Main Loop}
10:     for all $It = 1: It_{max}$ do
11:         for all $s = I: II$ do % Number of subpopulations%
12:             Perform SaDE on $Population_{s,It}$
13:             Combine the $Population_{s,It}$ with the best individual of other sub-population
14:                 Calculate fitness functions for combined individuals
15:                 $Population_{s,It+1} \leftarrow$ SaDE Selection Procedure on $Population_{s,It}$
16:                 Add $NFE_0$
17:         end for
18:         if mod($It_{Max}, \beta_1) = 0$ and $|BFS(It) - BFS(It-1)| < \delta$ then
19:             Do local search on the worst vector in PopulationI and add $NFE_0$
20:         end if
21:         if mod($It_{Max}, \beta_2) = 0$ then
22:             Calculate $R$ using: $R = 0.5f_{\mu}(x^*) + 0.5R_{old}$
23:         else
24:             $R = R_{old}$
25:         end if
26:         Select $(x^*, \mu^*)$
27:         $BFS(It+1) = x^*$
28:         if $|BFS(It) - BFS(It-1)| = 0$ then
29:             count = count + 1
30:         else
31:             count = 0
32:         end if
33:         if $NFE_0 > MaxNFE$ or count $> \beta_3$ then
34:             break
35:         end if
36:     end for
37:     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: $N_{pop_1} = 200$, $N_{pop_2} = 20$, $It_{Max} = 500$, $\mu_0 = -5$, $\beta_1 = 0.1 \times It_{Max}$, $\beta_2 = 5$, $\delta = 0.0001$, $\beta_3 = 0.02 \times It_{Max}$, $MaxNFE = 240000$.

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 CCiALF$_{NLS}$, which is compared directly with CCALF$_{NLS}$.

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].

$$|h_i(x)| - \epsilon \leq 0$$

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

$$\min_{x} f(x), \quad x \in \mathbb{R}^n$$

$$g_i(x) \geq 0, \quad i = 1, ..., m$$

$$h_i(x) + \epsilon \geq 0, \quad i = 1, ..., n$$

$$\epsilon - h_i(x) \geq 0, \quad i = 1, ..., n$$

$$lb_i \leq x_i \leq ub_i, \quad i = 1, ..., p$$

The CEC’2006 test functions are used to evaluate the performance of CCALF$_{NLS}$ and CCiALF$_{NLS}$ algorithms. For each problem 30 independent
runs are performed and mean (\(M_n\)) and standard deviation (\(sd\)) are provided in Table 3. In order to compare the performance of CCiALF\(_{NLS}\) with CCALF\(_{NLS}\) algorithm, the t-test with the significance level of 0.05 is performed. To perform the t-test the following hypotheses are considered.

\[
H_0 : M_{n_j} = M_{n_i} \\
H_a : M_{n_j} \neq M_{n_i}
\]

where \(j=\text{CCiALF}_{NLS}\) and \(i=\text{CCALF}_{NLS}\). After performing the pairwise t-test for both algorithms, if CCiALF\(_{NLS}\) is equal, superior or inferior to the compared CCALF\(_{NLS}\) algorithm, then \(h\) is equal to 0, 1 and -1 respectively. As shown in Table 3, CCiALF\(_{NLS}\) is superior to CCALF\(_{NLS}\) in 18 test functions (out of 22) and in 2 test functions both algorithms have the same performance. In addition, CCALF\(_{NLS}\) failed to obtain any feasible solution for g03 and g14 (denoted by “inf” in Table 3). Fig. 4 shows the convergence plots for CCiALF\(_{NLS}\) and CCALF\(_{NLS}\) 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, CCALF\(_{NLS}\) converged to better solutions and were faster than CCALF\(_{NLS}\). The aforementioned results demonstrate that CCiALF\(_{NLS}\) outperforms CCALF\(_{NLS}\) in terms of constraint handling effectiveness, solution quality and convergence speed.

Table 3: Comparison between CCiALF\(_{NLS}\) and CCALF\(_{NLS}\) in solving CEC’2006.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>g01</th>
<th>g02</th>
<th>g03</th>
<th>g04</th>
<th>g05</th>
<th>g06</th>
<th>g07</th>
<th>g08</th>
<th>g09</th>
<th>g10</th>
<th>g11</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCiALF(_{NLS})</td>
<td>Mn</td>
<td>-15</td>
<td>-0.789524</td>
<td>-0.92708</td>
<td>-30665.5</td>
<td>5184.802</td>
<td>-6961.81</td>
<td>24.30625</td>
<td>-0.09583</td>
<td>680.6301</td>
<td>7049.272</td>
</tr>
<tr>
<td></td>
<td>sd</td>
<td>1.23E-15</td>
<td>1.04E-02</td>
<td>1.99E-01</td>
<td>2.03E-05</td>
<td>1.13E+02</td>
<td>1.22E-02</td>
<td>1.70E-04</td>
<td>1.20E-17</td>
<td>1.16E-10</td>
<td>2.05E-06</td>
</tr>
<tr>
<td>CCALF(_{NLS})</td>
<td>Mn</td>
<td>11.46614</td>
<td>-8.48E+05</td>
<td>inf</td>
<td>-28091.3</td>
<td>5482.945</td>
<td>-3387.35</td>
<td>30.81E66</td>
<td>-0.66E+05</td>
<td>683.1372</td>
<td>19171.74</td>
</tr>
<tr>
<td></td>
<td>sd</td>
<td>7.78E-01</td>
<td>4.56E-02</td>
<td>inf</td>
<td>6.49E+02</td>
<td>3.93E+02</td>
<td>3.84E+02</td>
<td>3.84E+02</td>
<td>3.59E+02</td>
<td>6.55E-61</td>
<td>3.98E-03</td>
</tr>
<tr>
<td>h</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>g12</th>
<th>g13</th>
<th>g14</th>
<th>g15</th>
<th>g16</th>
<th>g17</th>
<th>g18</th>
<th>g19</th>
<th>g20</th>
<th>g21</th>
<th>g24</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCiALF(_{NLS})</td>
<td>Mn</td>
<td>-1</td>
<td>0.511189</td>
<td>-14.5182</td>
<td>9260931</td>
<td>-1.00516</td>
<td>5025.852</td>
<td>-0.821792</td>
<td>32.68655</td>
<td>22.05256</td>
<td>-3.584</td>
</tr>
<tr>
<td></td>
<td>sd</td>
<td>0.90E+06</td>
<td>2.46E-01</td>
<td>5.23E-01</td>
<td>1.56E+00</td>
<td>6.31E-08</td>
<td>5.59E+01</td>
<td>1.50E-01</td>
<td>5.09E+06</td>
<td>1.07E-02</td>
<td>7.86E+08</td>
</tr>
<tr>
<td>CCALF(_{NLS})</td>
<td>Mn</td>
<td>-0.61313</td>
<td>0.90E+068</td>
<td>inf</td>
<td>9637127</td>
<td>-1.10450</td>
<td>8985.492</td>
<td>-3.398792</td>
<td>37.09E75</td>
<td>545.5552</td>
<td>633.0741</td>
</tr>
<tr>
<td></td>
<td>sd</td>
<td>1.09E+02</td>
<td>2.02E-01</td>
<td>inf</td>
<td>2.20E+00</td>
<td>1.83E-01</td>
<td>1.01E+02</td>
<td>1.40E+01</td>
<td>3.76E+01</td>
<td>2.13E+02</td>
<td>2.83E+02</td>
</tr>
<tr>
<td>h</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

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
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.

<table>
<thead>
<tr>
<th>Function</th>
<th>Opt. value</th>
<th>P-Bf AFSP</th>
<th>COMDE</th>
<th>A-DDE</th>
<th>PSGA</th>
<th>APF-GA</th>
<th>M-ABC</th>
<th>CB-ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>h=-1</td>
<td>0.2000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NFE</td>
<td>124,312</td>
<td>200,000</td>
<td>180,000</td>
<td>100,000</td>
<td>500,000</td>
<td>83,500</td>
<td>198,270</td>
<td>240,000</td>
</tr>
<tr>
<td>h=-1</td>
<td>0.4000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NFE</td>
<td>124,312</td>
<td>200,000</td>
<td>180,000</td>
<td>100,000</td>
<td>500,000</td>
<td>83,500</td>
<td>198,270</td>
<td>240,000</td>
</tr>
<tr>
<td>h=-1</td>
<td>0.4000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NFE</td>
<td>124,312</td>
<td>200,000</td>
<td>180,000</td>
<td>100,000</td>
<td>500,000</td>
<td>83,500</td>
<td>198,270</td>
<td>240,000</td>
</tr>
</tbody>
</table>

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 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.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>-47.738</td>
<td>-47.748</td>
<td>-47.644</td>
<td>-47.769</td>
<td>-47.769</td>
</tr>
<tr>
<td>g14</td>
<td>-47.767</td>
<td>-47.748</td>
<td>-47.271</td>
<td>-47.769</td>
<td>-47.769</td>
</tr>
<tr>
<td>-47.765</td>
<td>3.74E-02</td>
<td>1.00E-04</td>
<td>2.46E-01</td>
<td>1.02E-05</td>
<td>4.04E-08</td>
</tr>
<tr>
<td>h</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NFE</td>
<td>100,000</td>
<td>500,000</td>
<td>N/A</td>
<td>239,715</td>
<td>961.715</td>
</tr>
<tr>
<td>g15</td>
<td>-47.271</td>
<td>-47.7648</td>
<td>-47.7649</td>
<td>-47.7649</td>
<td>-47.7649</td>
</tr>
<tr>
<td>961.715</td>
<td>1.74E-02</td>
<td>0.00E+00</td>
<td>1.42E-01</td>
<td>2.81E-11</td>
<td>8.65E-08</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NFE</td>
<td>100,000</td>
<td>500,000</td>
<td>189,000</td>
<td>135,180</td>
<td>961.715</td>
</tr>
<tr>
<td>g16</td>
<td>-47.567</td>
<td>N/A</td>
<td>-46.537</td>
<td>-47.7648</td>
<td>-47.7649</td>
</tr>
<tr>
<td>-1.905</td>
<td>4.68E-15</td>
<td>9.39E+01</td>
<td>7.90E-11</td>
<td>9.77E-09</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NFE</td>
<td>100,000</td>
<td>500,000</td>
<td>189,000</td>
<td>135,180</td>
<td>961.715</td>
</tr>
<tr>
<td>g17</td>
<td>961.769</td>
<td>961.7150</td>
<td>961.719</td>
<td>961.715</td>
<td>961.7150</td>
</tr>
<tr>
<td>b</td>
<td>-1.905</td>
<td>-1.905155</td>
<td>-1.905</td>
<td>-1.905</td>
<td>-1.905155</td>
</tr>
<tr>
<td>g18</td>
<td>-1.905</td>
<td>N/A</td>
<td>-1.905</td>
<td>-1.905</td>
<td>-1.905155</td>
</tr>
<tr>
<td>-1.905</td>
<td>8.14E+01</td>
<td>9.78E+00</td>
<td>7.90E-11</td>
<td>9.77E-09</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NFE</td>
<td>100,000</td>
<td>500,000</td>
<td>189,000</td>
<td>239,715</td>
<td>961.715</td>
</tr>
<tr>
<td>g19</td>
<td>961.769</td>
<td>961.7150</td>
<td>961.719</td>
<td>961.715</td>
<td>961.7150</td>
</tr>
<tr>
<td>b</td>
<td>-1.905</td>
<td>-1.905155</td>
<td>-1.905</td>
<td>-1.905</td>
<td>-1.905155</td>
</tr>
<tr>
<td>g20</td>
<td>961.769</td>
<td>961.7150</td>
<td>961.719</td>
<td>961.715</td>
<td>961.7150</td>
</tr>
<tr>
<td>b</td>
<td>-1.905</td>
<td>-1.905155</td>
<td>-1.905</td>
<td>-1.905</td>
<td>-1.905155</td>
</tr>
<tr>
<td>g21</td>
<td>-1.905</td>
<td>N/A</td>
<td>-1.905</td>
<td>-1.905</td>
<td>-1.905155</td>
</tr>
<tr>
<td>-1.905</td>
<td>8.14E+01</td>
<td>9.78E+00</td>
<td>7.90E-11</td>
<td>9.77E-09</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NFE</td>
<td>100,000</td>
<td>500,000</td>
<td>189,000</td>
<td>239,715</td>
<td>961.715</td>
</tr>
<tr>
<td>g22</td>
<td>961.769</td>
<td>961.7150</td>
<td>961.719</td>
<td>961.715</td>
<td>961.7150</td>
</tr>
<tr>
<td>b</td>
<td>-1.905</td>
<td>-1.905155</td>
<td>-1.905</td>
<td>-1.905</td>
<td>-1.905155</td>
</tr>
<tr>
<td>g23</td>
<td>-1.905</td>
<td>N/A</td>
<td>-1.905</td>
<td>-1.905</td>
<td>-1.905155</td>
</tr>
<tr>
<td>-1.905</td>
<td>8.14E+01</td>
<td>9.78E+00</td>
<td>7.90E-11</td>
<td>9.77E-09</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NFE</td>
<td>100,000</td>
<td>500,000</td>
<td>189,000</td>
<td>239,715</td>
<td>961.715</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Superior</td>
<td>13(100%)</td>
<td>4(31%)</td>
<td>4(31%)</td>
<td>10(73%)</td>
<td>8(37%)</td>
<td>16(73%)</td>
<td>5(23%)</td>
</tr>
<tr>
<td>Equal</td>
<td>0(0%)</td>
<td>8(61%)</td>
<td>8(61%)</td>
<td>10(42%)</td>
<td>10(42%)</td>
<td>5(23%)</td>
<td>14(61%)</td>
</tr>
<tr>
<td>Inferior</td>
<td>0(0%)</td>
<td>1(8%)</td>
<td>1(8%)</td>
<td>0(0%)</td>
<td>4(18%)</td>
<td>1(4%)</td>
<td>3(14%)</td>
</tr>
<tr>
<td>Total functions</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
</tr>
</tbody>
</table>

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
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 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\cite{5} are used here to obtain the results shown in Table 7. In this case, CCIaLF algorithm has similar or 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 \cite{34,41,4,15} and \cite{35}. The first problem is known as welded beam design \cite{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 \cite{8}:

\[
\begin{align*}
\min & \quad f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \\
\text{s.t.} & \quad g_1(x) \equiv \tau_{\text{max}} - \tau(x) \geq 0 \\
& \quad g_2(x) \equiv \sigma_{\text{max}} - \sigma(x) \geq 0 \\
& \quad g_3(x) \equiv x_4 - x_1 \geq 0 \\
& \quad g_4(x) \equiv 5.0 - 0.10471x_1^2 - 0.04811x_3x_4(14.0 + x_2) \geq 0 \\
& \quad g_5(x) \equiv x_1 - 0.125 \geq 0 \\
& \quad g_6(x) \equiv \delta_{\text{max}} - \delta(x) \geq 0 \\
& \quad g_7(x) \equiv P_e(x) - 6000 \geq 0 \\
& \quad 0.1 \leq (x_1, x_4) \leq 2, \quad 0.1 \leq (x_2, x_3) \leq 10 \\
\end{align*}
\]

where,
\[
\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''x_2^2 + (\tau'')^2}, \quad \tau' = \frac{P}{\sqrt{2x_1x_2}}, \quad \tau'' = \frac{MR}{J}
\]

\[
M = P\left(L + \frac{x_2}{2}\right), \quad R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}
\]

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

\[
\delta(x) = \frac{4PL^3}{Ex_3^3x_4}, \quad P_e(x) = \frac{4.013E\sqrt{\frac{x_2^2x_4^2}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)
\]

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:

\[x_{CCiALF} = (0.205729639785972, 3.470488665631285, 9.036623910355711, 0.205729639786304)\]

The reported decision variable is feasible and constraints are:

\[g_1(x) = 7.24E - 10, \quad g_2(x) = 2.00E - 08, \quad g_3(x) = 3.32E - 13, \quad g_4(x) = 3.432983785, \quad g_5(x) = 0.08072964, \quad g_6(x) = 0.235540323 \text{ and } g_7(x) = 1.88E - 08\]

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:

\[x_{CCiALF} = (0.8125, 0.4375, 42.098445595854810, 176.6365958424410)\]

and the constraints are \(g_1(x) = 0, \quad g_2(x) = 3.59E - 02, \quad g_3(x) = 0 \text{ and } g_4(x) = 63.36\) which demonstrates the obtained solution is feasible.
Table 8: Results for engineering problems (Note that “N/A” denotes the results are not available and the best mean values are presented in boldface).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Welded beam design</th>
<th>Pressure vessel design</th>
<th>Weight of a tension/compression string</th>
<th>Speed reducer design</th>
<th>Three-bar truss design</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>best mean worst std NFE h</td>
<td>best mean worst std NFE h</td>
<td>best mean worst std NFE h</td>
<td>best mean worst std NFE h</td>
<td>best mean worst std NFE h</td>
</tr>
<tr>
<td>Coe02 [8]</td>
<td>1.728226 1.792654 1.993408 7.47E-02 80,000 1</td>
<td>6059.946341 6177.253268 6469.32201 1.31E+02 80,000 1</td>
<td>0.012679 0.012742 0.012973 5.90E-05 80,000 1</td>
<td>COMDE [27] 2994.4710661 2994.4710661 2994.4710661 1.54E-12 21,000 0</td>
<td></td>
</tr>
<tr>
<td>COMDE [27]</td>
<td>1.724852 1.724852 1.724852 1.60E-12 20,000 0</td>
<td>HCP [29] 6059.69 6171.13 N/A 1.13E+02 30,000 1</td>
<td>0.012679 0.012742 0.012973 5.90E-05 80,000 1</td>
<td>MVDE [11] 2994.471066 2994.471066 2994.471066 2.82E-07 30,000 0</td>
<td></td>
</tr>
<tr>
<td>MVDE [11]</td>
<td>1.7248527 1.7248621 1.7249215 7.88E-06 15,000 1</td>
<td>CB-ABC [5] 6059.714387 6059.997236 6090.53528 2.91E+00 15,000 0</td>
<td>0.012679 0.012742 0.012973 5.90E-05 80,000 1</td>
<td>CB-ABC [5] 2994.471066 2994.471066 2994.471066 2.48E-07 15,000 0</td>
<td></td>
</tr>
<tr>
<td>CB-ABC [5]</td>
<td>1.724852 1.724852 N/A 2.38E-11 15,000 0</td>
<td>CCiALF 6059.714335 6059.714335 6059.714335 1.01E-11 12,000 0</td>
<td>0.012679 0.012742 0.012973 5.90E-05 80,000 1</td>
<td>CCiALF 2994.471066 2994.471066 2994.471066 2.31E-12 10,000 0</td>
<td></td>
</tr>
<tr>
<td>CCiALF</td>
<td>1.724852 1.724852 1.724852 5.11E-07 10,000 0</td>
<td></td>
<td>0.012679 0.012742 0.012973 5.90E-05 80,000 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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]:

21
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]:

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

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:

\[
\begin{align*}
\mathbf{x} &= (0.051689059696547, 0.356717706450724, 11.288967706745048), \quad g_1(\mathbf{x}) = 2.22E - 16, \quad g_2(\mathbf{x}) = 1.11E - 16, \quad g_3(\mathbf{x}) = 4.05 \quad \text{and} \quad g_4(\mathbf{x}) = 7.28E - 01.
\end{align*}
\]

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

\[
\begin{align*}
\min & \quad f(\mathbf{x}) = (x_3 + 2)x_2x_1^2 \\
\text{s.t.} & \quad g_1(\mathbf{x}) = \frac{x_3^3x_4}{71785x_1^4} - 1 \geq 0 \\
& \quad g_2(\mathbf{x}) = 1 - \frac{4x_2^2 - x_1x_2}{12566(x_2x_3^2 - x_1^3)} - \frac{1}{5108x_1^2} \geq 0 \\
& \quad g_3(\mathbf{x}) = \frac{140.45x_1}{x_2^2x_3} - 1 \geq 0 \\
& \quad g_4(\mathbf{x}) = 1 - \frac{x_1 + x_2}{1.5} \geq 0 \\
& \quad 0.05 \leq x_1 \leq 2, \quad 0.25 \leq x_2 \leq 1.3, \quad 2 \leq x_3 \leq 15
\end{align*}
\]
of speed reducer. The speed reducer problem has seven decision variables and eleven constraints. The formula of speed reducer design is as follows:

$$\min f(x) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1$$

$$\left(x_6^2 + x_7^2\right) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_6^2 + x_5x_7^2)$$

s.t.

$$g_1(x) = 1 - \frac{27}{x_1x_2^2x_3} \geq 0$$

$$g_2(x) = 1 - \frac{397.5}{x_1x_2^2x_3} \geq 0$$

$$g_3(x) = 1 - \frac{1.93x_4^3}{x_2x_3x_6^2} \geq 0$$

$$g_4(x) = 1 - \frac{1.93x_5^3}{x_2x_3x_7^2} \geq 0$$

$$g_5(x) = 1 - \frac{1}{110x_6^3}\sqrt{\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6} \geq 0$$

$$g_6(x) = 1 - \frac{1}{85x_7^3}\sqrt{\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5 \times 10^6} \geq 0$$

$$g_7(x) = 1 - \frac{x_2x_3}{40} \geq 0$$

$$g_8(x) = 1 - \frac{5x_2}{x_1} \geq 0$$

$$g_9(x) = 1 - \frac{x_1}{12x_2} \geq 0$$

$$g_{10}(x) = 1 - \frac{(1.5x_6 + 1.9)}{x_4} \geq 0$$

$$g_{11}(x) = 1 - \frac{(1.1x_7 + 1.9)}{x_5} \geq 0$$

$$2.6 \leq x_1 \leq 3.6, \quad 0.7 \leq x_2 \leq 0.8, \quad 17 \leq x_3 \leq 28, \quad 7.3 \leq x_4 \leq 8.3$$

$$7.8 \leq x_5 \leq 8.3, \quad 2.9 \leq x_6 \leq 3.9, \quad 5 \leq x_7 \leq 5.5$$

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:
\[ x = (3.505071907646737, 0.700070905298447, 17.000572946016090, 7.41363851442300, 7.824441935611341, 3.35267747256310, 5.287622983298447, 3.1700057294601609, 7.413638851442300, 7.824441935611341), \]

\[ g_1(x) = 7.39E-02, \quad g_2(x) = 1.98E-01, \quad g_3(x) = 4.92E-01, \quad g_4(x) = 9.4E-01, \quad g_5(x) = 7.79E-12, \quad g_6(x) = 5.65E-12, \quad g_7(x) = 7.02E-01, \quad g_8(x) = 3.27E-12, \quad g_9(x) = 5.83E-01, \quad g_{10}(x) = 5.13E-02 \]

and \[ g_{11}(x) = 1.08E-11 \] 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.

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

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: \[ x = (0.788675131095601, 0.408248300361130) \], \[ g_1(x) = 0, \quad g_2(x) = 1.46E+00 \] and \[ g_3(x) = 5.36E-01 \] 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 co-evolutionary 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, CCiALF\textsubscript{NLS} and CCALF\textsubscript{NLS} algorithms are applied to solve the CEC’2006 test functions. The experimental results have shown that CCiALF\textsubscript{NLS} significantly outperforms CCALF\textsubscript{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


