Constrained Evolutionary Optimization by Means of $(\mu + \lambda)$-Differential Evolution and Improved Adaptive Trade-Off Model

Yong Wang ywang@csu.edu.cn
School of Information Science and Engineering, Central South University, Changsha 410083, People's Republic of China

Zixing Cai zxcai@csu.edu.cn
School of Information Science and Engineering, Central South University, Changsha 410083, People's Republic of China

Abstract

This paper proposes a $(\mu + \lambda)$-differential evolution and an improved adaptive trade-off model for solving constrained optimization problems. The proposed $(\mu + \lambda)$-differential evolution adopts three mutation strategies (i.e., rand/1 strategy, current-to-best/1 strategy, and rand/2 strategy) and binomial crossover to generate the offspring population. Moreover, the current-to-best/1 strategy has been improved in this paper to further enhance the global exploration ability by exploiting the feasibility proportion of the last population. Additionally, the improved adaptive trade-off model includes three main situations: the infeasible situation, the semi-feasible situation, and the feasible situation. In each situation, a constraint-handling mechanism is designed based on the characteristics of the current population. By combining the $(\mu + \lambda)$-differential evolution with the improved adaptive trade-off model, a generic method named $(\mu + \lambda)$-constrained differential evolution ($(\mu + \lambda)$-CDE) is developed. The $(\mu + \lambda)$-CDE is utilized to solve 24 well-known benchmark test functions provided for the special session on constrained real-parameter optimization of the 2006 IEEE Congress on Evolutionary Computation (CEC2006). Experimental results suggest that the $(\mu + \lambda)$-CDE is very promising for constrained optimization, since it can reach the best known solutions for 23 test functions and is able to successfully solve 21 test functions in all runs. Moreover, in this paper, a self-adaptive version of $(\mu + \lambda)$-CDE is proposed which is the most competitive algorithm so far among the CEC2006 entries.

Keywords

Constrained optimization problems, constraint-handling technique, evolutionary algorithm, differential evolution, adaptive trade-off model.

1 Introduction

Constrained optimization problems (COPs) belong to a kind of mathematical programming problems which are frequently encountered in the disciplines of science and engineering. Solving COPs is also an important area in the optimization field. This

paper considers the following general single-objective COPs:

$$\begin{cases} \min f(\vec{x}), & \vec{x} = (x_1, x_2, \ldots, x_n) \in \Re^n \\ g_j(\vec{x}) \le 0, & j = 1, \ldots, l \\ h_j(\vec{x}) = 0, & j = l+1, \ldots, p \\ l_i \le x_i \le u_i & 1 \le i \le n \end{cases} \tag{1}$$

where $(x_1, x_2, \ldots, x_n)$ are the n decision variables; $f(\vec{x})$ is the objective function; $g_j(\vec{x})$ is the jth inequality constraint; $h_j(\vec{x})$ is the jth equality constraint; and l_i and u_i are the lower and upper bounds of the ith decision variable x_i.

The search space S is an n-dimensional rectangular space in $\Re^n$ defined by the parametric constraints:

$$l_i \le x_i \le u_i, \quad 1 \le i \le n. \tag{2}$$

The feasible region Ω is defined by the l inequality constraints $g_j(\vec{x})$ and the $(p-l)$ equality constraints $h_j(\vec{x})$. Any point $\vec{x} \in \Omega$ is called a feasible solution; otherwise, $\vec{x}$ is an infeasible solution.

If an inequality constraint satisfies $g_j(\vec{x}) = 0$ ($j \in \{1, \ldots, l\}$) at any point $\vec{x} \in \Omega$, we say it is *active* at $\vec{x}$. All equality constraints $h_j(\vec{x})$ ($j = l+1, \ldots, p$) are considered active at all points of Ω.

As for COPs, equality constraints are always transformed into inequality constraints as

$$|h_j(\vec{x})| - \delta \le 0, \tag{3}$$

where $j = l+1, \ldots, p$ and δ is a positive tolerance value. Usually, the degree of constraint violation of individual $\vec{x}$ on the jth constraint is calculated as follows:

$$G_j(\vec{x}) = \begin{cases} \max\{0, g_j(\vec{x})\}, & 1 \le j \le l \\ \max\{0, |h_j(\vec{x})| - \delta\}, & l+1 \le j \le p \end{cases}. \tag{4}$$

COPs often have to deal with linear/nonlinear and equality/inequality constraints, multimodal objective functions, concave feasible regions, and so on. Consequently, many optimization methods may not be attractive since they easily get stuck at local optima or are computationally expensive. In the past decade, evolutionary algorithms (EAs) have gained more and more attention for solving COPs due to their flexibility and adaptability to the task at hand. As a result, a large number of constrained optimization evolutionary algorithms (COEAs) have been proposed (Michalewicz and Schoenauer, 1996; Coello, 2002; Wang et al., 2008).

One of the major issues for COEAs is how to handle the constraints since EAs are unconstrained optimization methods. Some techniques to handle constraints in EAs have been proposed. Michalewicz and Schoenauer (1996) and Coello (2002) provided an extensive survey of constraint-handling techniques suitable for EAs and grouped them into four and five categories, respectively. However, the most popular constraint-handling techniques at present are methods based on penalty functions (Coit et al., 1996; Tessema and Yen, 2009), methods based on preference of feasible solutions over infeasible solutions (Deb, 2000; Mezura-Montes and Coello, 2005), and methods based on multiobjective optimization techniques (Venkatraman and Yen, 2005; Cai and Wang,

2006). The principal idea of methods based on penalty functions is to consider a COP as unconstrained by introducing a penalty term into the original objective function to penalize those individuals that violate constraints. As for methods based on preference of feasible solutions over infeasible solutions, feasible solutions are always considered better than infeasible solutions to a certain degree. The main features of methods based on multiobjective optimization techniques are twofold: (1) to convert a COP into an unconstrained multiobjective optimization problem; and (2) to utilize multiobjective optimization techniques for the converted problem.

On the other hand, it is very important to develop an effective search algorithm that can find global optimal solutions for COPs. As an important branch of EAs, differential evolution (DE) was originally introduced by Storn and Price (1995) and grew out of Price's attempts to solve the Chebychev polynomial fitting problem. Similar to other EAs, DE is a population-based optimization algorithm that can solve hard optimization problems, but it is simpler and easier to implement. DE was originally designed for unconstrained optimization problems. However, being fascinated by the prospect and potential of DE, many researchers have applied DE to solve COPs.

As stated above, the constraint-handling technique and search algorithm are two very important aspects when solving COPs. However, it is noteworthy that when using DE to cope with COPs, the commonly used constraint-handling technique is the feasibility-based criteria proposed by Deb (2000). In addition, the conventional DE for COPs applies an exclusively knock-out selection mechanism for survival. Thus, the constraint-handling capability and search performance of these methods are not good.

Unlike the previous methods, this paper proposes a $(\mu + \lambda)$-constrained differential evolution ($(\mu + \lambda)$-CDE) which combines a $(\mu + \lambda)$-DE with an improved adaptive trade-off model to deal with COPs. In the approach proposed, each individual in the population is applied to generate three offspring by making use of three well-known mutation strategies (i.e., the rand/1 strategy, the current-to-best/1 strategy, and the rand/2 strategy) and binomial crossover of DE. As a result, an offspring population is obtained. Moreover, the current-to-best/1 strategy is further revised to enhance global search capability. After combining the parent and offspring populations, a selection scheme similar to the $(\mu + \lambda)$-evolution strategy (ES) is utilized to choose potential individuals for survival. In this paper, the constraint-handling technique is based on an improved adaptive trade-off model (IATM). Following its predecessor, that is, the adaptive trade-off model (ATM) proposed by the authors recently (Wang et al., 2008), IATM also contains three main situations: the infeasible situation, the semi-feasible situation, and the feasible situation. Furthermore, a constraint-handling mechanism is designed for a single situation. By combining the $(\mu + \lambda)$-DE with the IATM, the method proposed in this paper has the concrete capability of handling the constraints and searching for the optimal solutions.

The effectiveness and robustness of the proposed $(\mu + \lambda)$-CDE are evaluated on 24 well-known benchmark test functions presented for the CEC2006 special session on constrained real-parameter optimization. The empirical evidence indicates that the overall performance achieved by the $(\mu + \lambda)$-CDE is quite good since the $(\mu + \lambda)$-CDE can reach the best known solutions for 23 test functions and achieve a 100% success rate for 21 test functions.

The remainder of this paper is organized as follows. In Section 2, DE is briefly introduced. Section 3 introduces the related work. Section 4 presents a detailed description of the $(\mu + \lambda)$-CDE. In Section 5, the $(\mu + \lambda)$-CDE is exploited to produce experimental results for 24 standard benchmarks. Meanwhile, the results of the $(\mu + \lambda)$-CDE are compared with the ones from some relational literature. Section 6 is devoted to the

discussion of the effect of the mechanisms proposed in this paper on performance. Finally, the paper concludes in Section 7.

2 Differential Evolution

Differential evolution (DE) is a very popular EA and exhibits remarkable performance in optimizing a wide variety of problems in diverse fields. Usually, DE includes three main parameters: scaling factor F, crossover control parameter CR, and population size NP. During evolution, the population of DE contains NP n-dimensional individuals:

$$\vec{x}_{i,t} = \{x_{i,1,t}, x_{i,2,t}, \ldots, x_{i,n,t}\}, \quad i = 1, 2, \ldots, NP \tag{5}$$

where t denotes the generation number. The initial population should ideally cover the entire search space by uniformly and randomly distributing each variable of an individual between the lower and upper bounds predefined for each variable.

At each generation, DE employs the mutation and crossover operations to produce a trial vector $\vec{u}_{i,t}$ for each individual $\vec{x}_{i,t}$ (also called a target vector) in the current population. Thereafter, a selection operation is executed between the trial vector $\vec{u}_{i,t}$ and the target vector $\vec{x}_{i,t}$.

2.1 Mutation Operation

In the mutation operation, a mutant vector $\vec{v}_{i,t}$ is generated for each target vector $\vec{x}_{i,t}$ by making use of a mutation strategy. During the past decade, various mutation strategies have been proposed. The following are five mutation strategies frequently used in the literature:

- rand/1:

$$\vec{v}_{i,t} = \vec{x}_{r1,t} + F \cdot (\vec{x}_{r2,t} - \vec{x}_{r3,t}) \tag{6}$$

- best/1:

$$\vec{v}_{i,t} = \vec{x}_{\text{best},t} + F \cdot (\vec{x}_{r1,t} - \vec{x}_{r2,t}) \tag{7}$$

- current-to-best/1:

$$\vec{v}_{i,t} = \vec{x}_{i,t} + F \cdot (\vec{x}_{\text{best},t} - \vec{x}_{i,t}) + F \cdot (\vec{x}_{r1,t} - \vec{x}_{r2,t}) \tag{8}$$

- best/2:

$$\vec{v}_{i,t} = \vec{x}_{\text{best},t} + F \cdot (\vec{x}_{r1,t} - \vec{x}_{r2,t}) + F \cdot (\vec{x}_{r3,t} - \vec{x}_{r4,t}) \tag{9}$$

- rand/2:

$$\vec{v}_{i,t} = \vec{x}_{r1,t} + F \cdot (\vec{x}_{r2,t} - \vec{x}_{r3,t}) + F \cdot (\vec{x}_{r4,t} - \vec{x}_{r5,t}) \tag{10}$$

where indices $r1$, $r2$, $r3$, $r4$, and $r5$ are distinct integers randomly selected from the set $\{1, 2, \ldots, NP\}\backslash\{i\}$ and $\vec{x}_{\text{best},t}$ is the best individual in the population at generation t. The scaling factor F is a real number and is usually chosen between 0 and 1.

2.2 Crossover Operation

After mutation, a binomial crossover operation is applied to create the trial vector $\vec{u}_{i,t}$ as follows:

$$u_{i,j,t} = \begin{cases} v_{i,j,t} & \text{if rand}_j(0,1) \leq CR \quad \text{or } j = j_{\text{rand}} \\ x_{i,j,t} & \text{otherwise} \end{cases} \tag{11}$$

where $i = 1, 2, \ldots, NP$, $j = 1, 2, \ldots, n$, j_{rand} is an integer randomly chosen from 1 to n, and $\text{rand}_j(0, 1)$ is a uniformly distributed random number on the interval [0,1] which is newly generated for each j. The parameter j_{rand} is adopted to ensure that the trial vector $\vec{u}_{i,t}$ differs from its target vector $\vec{x}_{i,t}$ by at least one parameter. The crossover control parameter CR usually ranges between 0 and 1.

2.3 Selection Operation

In the selection operation, the target vector $\vec{x}_{i,t}$ is compared against the trial vector $\vec{u}_{i,t}$ in terms of the objective function value, and the better one will be added to the next population:

$$\vec{x}_{i,t+1} = \begin{cases} \vec{u}_{i,t} & \text{if } f(\vec{u}_{i,t}) \leq f(\vec{x}_{i,t}) \\ \vec{x}_{i,t} & \text{otherwise} \end{cases} \tag{12}$$

3 Related Work

Next, some related work using DE to solve COPs will be briefly discussed.

Storn (1999) proposed constraint adaptation in which all the constraints of the problems are relaxed at the beginning so that all the individuals in the initial population are feasible and the constraints are gradually tightened until they arrive at the original constraints. Lampinen and Zelinka (1999a,b,c) combined a penalty approach with DE to solve engineering design problems. Lin et al. (2002) proposed the use of an augmented Lagrangian function to guide the search. Lampinen (2002) extended DE to handle nonlinear constraint functions by modifying the selection operation of DE. Runarsson and Yao (2005) proposed an improved stochastic ranking method. This method contains a differential variation operator, which resembles the mutation operator of DE. Mezura-Montes et al. (2005) incorporated a diversity mechanism into DE. The diversity mechanism enables infeasible individuals with promising objective function values to remain in the next population. Takahama and Sakai (2006) proposed a method called εDE, which applies the ε constrained method to DE. Huang et al. (2006) introduced a self-adaptive DE for COPs. During the evolution, the suitable trial vector generation strategies and parameter settings are gradually self-adapted by learning from their previous experience. Tasgetiren and Suganthan (2006) presented a multi-populated DE. In this method, each subpopulation conducts its search in parallel. Moreover, the subpopulations are periodically regrouped to provide information exchange. Kukkonen and Lampinen (2006) exploited a generalized DE to solve COPs. Brest et al. (2006) proposed an extension of the self-adaptive DE, in which three DE mutation strategies are applied and the control parameters F and CR are self-adapted. Mezura-Montes et al. (2006) presented a modified DE for COPs. In this method, a new mutation operator is designed which combines information on both the best solution and the current parent to find new search directions. DE, coupled with a cultural algorithm, is proposed

by Becerra and Coello (2006). Huang et al. (2007) presented a co-evolutionary DE for constrained optimization. Recently, the book edited by Mezura-Montes (2009) included the most recent advances on nature-inspired algorithms to solve COPs. In this book, Takahama and Sakai (2009) proposed an improved εDE in which faster reduction of the relaxation of equality constraints and a higher rate of gradient-based mutation are adopted. Brest (2009) proposed an ε-self-adaptive DE which employs the ε constrained method similar to Takahama and Sakai (2006) to handle constraints. In this method, several mutation strategies are used and a self-adaptive mechanism is applied to the control parameters F and CR. Note that this method reduces the population size during the evolution. Mezura-Montes and Palomeque-Ortiz (2009) presented a self-adaptive diversity DE, which is an improved version of the one developed by Mezura-Montes et al. (2005). Among the four parameters of this method, three are self-adapted and the fourth is controlled deterministically. The behavior of these parameters and the online performance of this method have been analyzed. Very recently, Mallipeddi and Suganthan (2010) proposed an ensemble of constraint-handling techniques to solve COPs. In this method, each constraint-handling technique has its own population. As an instantiation of this method, DE is combined with four constraint-handling techniques.

Although the proposed algorithm in this paper also attempts to solve COPs by taking advantage of DE, its methodology is completely different from the above work; it attempts to combine a $(\mu + \lambda)$-DE with an improved adaptive trade-off model (IATM). Unlike the above methods, this paper adopts the IATM to deal with constraints and the $(\mu + \lambda)$-DE to search for the global optimum and combines them in an effective manner.

4 $(\mu + \lambda)$-Constrained Differential Evolution

In this section, the proposed $(\mu + \lambda)$-constrained differential evolution ($(\mu + \lambda)$-CDE) is introduced in detail from two aspects: $(\mu + \lambda)$-DE and IATM.

4.1 $(\mu + \lambda)$-DE

The $(\mu + \lambda)$-DE is similar to a $(\mu + \lambda)$-ES. In the $(\mu + \lambda)$-DE, μ parents (i.e., μ target vectors) are used to generate λ offspring (i.e., λ trial vectors). Thereafter, these μ parents and λ offspring are combined to select μ potential individuals for the next generation. This process is repeated generation after generation until some specific stopping criterion is met.

In the $(\mu + \lambda)$-DE, each parent produces three offspring by making use of three mutation strategies (i.e., rand/1, current-to-best/1, and rand/2) and binomial crossover of DE. Note that one mutation strategy is used to generate one mutant vector. As discussed previously, there are usually five mutation strategies in DE. Both the best/1 strategy and the best/2 strategy exploit the information of the best individual found so far in the population explicitly. Since the $(\mu + \lambda)$-DE directly adopts an elitist mechanism, if these two strategies are integrated into the $(\mu + \lambda)$-DE, the knowledge of the best individual will be reproduced promptly. Under this condition, the population might be easily trapped in a local optimum. Thus, only three representatives of the five mutation strategies are used in this study.

In addition, the current-to-best/1 strategy also relies on the best individual found so far. As a result, this strategy has a fast convergence speed and performs well when

solving unimodal problems. However, when solving multimodal problems, it runs the risk of getting stuck at a local optimum. In order to further enhance the global search capability and add the diversity of the population, the current-to-best/1 strategy has been improved as shown in Algorithm 1 where φ denotes the feasibility proportion of the last population.

if $rand < \varphi$ **then** // $rand$ is a uniformly distributed random number between 0 and 1
  $\vec{v}_{i,t} = \vec{x}_{i,t} + rand \cdot (\vec{x}_{r1,t} - \vec{x}_{i,t}) + F \cdot (\vec{x}_{r2,t} - \vec{x}_{r3,t})$;
  // current-to-rand/1 strategy
else
  $\vec{v}_{i,t} = \vec{x}_{i,t} + rand \cdot (\vec{x}_{\text{best},t} - \vec{x}_{i,t}) + F \cdot (\vec{x}_{r1,t} - \vec{x}_{r2,t})$;
  // current-to-best/1 strategy
end
if $rand < p_m$ **then** // p_m denotes the mutation probability
  execute the improved BGA mutation for $\vec{v}_{i,t}$;
end

Algorithm 1: The implementation of the improved current-to-best/1 strategy.

The improved current-to-best/1 strategy has the following features:

- As shown in Algorithm 1, the implementation of the improved current-to-best/1 strategy is based on the feasibility proportion φ of the last population. For COPs, at the early stage, the population might contain infeasible solutions only (i.e., $\varphi = 0$). In this case, it is reasonable for the individuals in the population to follow the best individual (i.e., the individual with the lowest degree of constraint violation) found so far for the purpose of approaching or entering the feasible region promptly. Thus, the current-to-best/1 strategy is utilized for the above purpose. Along with the evolution, more and more individuals in the population will become feasible, which means that the value of φ increases gradually during the evolution. Under this condition, if all the individuals in the population still learn the experience directly from the best individual, the diversity of the population will drastically decrease, and the population might easily converge to a local optimum. Therefore, the current-to-best/1 strategy is switched into the current-to-rand/1 strategy according to the feasibility proportion of the last population. It is necessary to note that the only difference between the current-to-best/1 strategy and the current-to-rand/1 strategy is that the current individual of the latter learns the information from other individuals chosen from the population at random. The main aim of the use of the current-to-rand/1 strategy is to enhance the global exploration ability of the population.

- The first scaling factor is uniformly chosen between 0 and 1 for both the current-to-best/1 strategy and the current-to-rand/1 strategy as shown in Algorithm 1, by which the method has a better exploration capability.

- In order to add the diversity of the population, the improved BGA mutation (Wang et al., 2007) is applied to the mutant vector $\vec{v}_{i,t}$ with a probability p_m. The

improved BGA mutation works as follows:

$$v_{i,j,t} = \begin{cases} v_{i,j,t} \pm rang_i \cdot \sum_{k=0}^{15} \alpha_k 2^{-k} & \text{if } rand < 1/n \\ v_{i,j,t} & \text{otherwise} \end{cases}, \quad j = 1, \ldots, n \tag{13}$$

where rand is a uniformly distributed random number between 0 and 1, rang_i defines the mutation range, and it is set to $(u_i - l_i) \cdot (1 - \text{current\_gen}/\text{total\_gen})^6$ where current_gen represents the current generation number and total_gen represents the total generation number. The + or − sign is chosen with a probability of 0.5 and $\alpha_k \in \{0, 1\}$ is randomly generated with $P(\alpha_k = 1) = 1/16$.

- The crossover operation of DE is not applied to the mutant vector created by the improved current-to-best/1 strategy, which is equivalent to the condition that the crossover control parameter *CR* is equal to 1.0. In this case, the trial vector is equal to the mutant vector, and the trial vector does not inherit any information from the target vector. As a result, the diversity of the population can be kept. In addition, the improved current-to-best/1 strategy is a rotation-invariant process if the crossover operation is neglected, which is useful for solving rotated problems.

In our method, if the jth element $v_{i,j,t}$ of the mutant vector $\vec{v}_{i,t}$ generated by the $(\mu + \lambda)$-DE violates the boundary constraint, this element is reflected back from the violated boundary using the following rule (Kukkonen and Lampinen, 2006):

$$v_{i,j,t} = \begin{cases} 2l_j - v_{i,j,t} & \text{if } v_{i,j,t} < l_j \\ 2u_j - v_{i,j,t} & \text{if } v_{i,j,t} > u_j \\ v_{i,j,t} & \text{otherwise} \end{cases} \tag{14}$$

4.2 Improved Adaptive Trade-Off Model (IATM)

In the $(\mu + \lambda)$-CDE, initially, a parent population P_t of size μ is generated (note that in this paper $\mu = NP$). Then the $(\mu + \lambda)$-DE is used to create an offspring population Q_t of size λ. A combined population $(P_t + Q_t)$ is formed by combining the parent population with the offspring population. The combined population $(P_t + Q_t)$ is of size $(\mu + \lambda)$.

In general, for constrained optimization, the combined population $(P_t + Q_t)$ may inevitably experience three situations: (1) the population contains infeasible individuals only (infeasible situation); (2) the population consists of both feasible and infeasible individuals (semi-feasible situation); and (3) the population is entirely composed of feasible individuals (feasible situation). In this paper, one constraint-handling mechanism is designed for one situation.

When determining whether a population is infeasible, semi-feasible, or feasible, we should consider the feasibility of the individuals in the population. The judgment of the feasibility of an individual is usually based on the degree of its constraint violation. In this paper, we employ two criteria to compute the degree of constraint violation of an individual.

As for certain kinds of COPs, the measures for the different constraint violations may have largely different scales because of the property of the constraints. Under this

condition, each constraint violation is normalized by dividing it by the largest violation of that constraint in the population, with the purpose of allotting equal importance to each constraint. First, the maximum constraint violation of each constraint is found by Equation (15):

$$G_j^{\max} = \max_{i=1,\ldots,(\mu+\lambda)} G_j(\vec{x}_i), \quad j \in \{1, \ldots, p\} \tag{15}$$

Then, the degree of constraint violation of each individual can be obtained by calculating the mean of the normalized constraint violations:

$$G(\vec{x}_i) = \frac{\sum_{j=1}^{p} G_j(\vec{x}_i)/G_j^{\max}}{p}, \quad i \in \{1, \ldots, (\mu + \lambda)\} \tag{16}$$

The differences among the constraints may not be significant when calculating the degree of constraint violation of an individual for certain kinds of COPs. In this case, the differences among the constraints should be emphasized; and therefore, we use the following criterion to measure the degree of constraint violation of an individual:

$$G(\vec{x}_i) = \sum_{j=1}^{p} G_j(\vec{x}_i), \quad i \in \{1, \ldots, (\mu + \lambda)\} \tag{17}$$

The above two criteria should be used based on the problems' characteristics. We make use of $G_j^{\max}$ of the initial population to measure the difference of the constraints. If $\max_{j=1,\ldots,p} G_j^{\max} - \min_{j=1,\ldots,p} G_j^{\max} > \xi$ where ξ is a user-defined parameter, then the difference of the constraints is significant, and the first criterion is used; otherwise, the difference of the constraints is not significant, and the second criterion is used. The criterion adopted should be based on the initial population and should be fixed during the evolution.

4.2.1 The Constraint-Handling Mechanism for the Infeasible Situation

In this situation, since the combined population $(P_t + Q_t)$ contains no feasible solutions, the aim is to promptly motivate the population toward the feasible region.

Intuitively, with respect to constrained optimization, it does not make sense if an individual is far away from the boundaries of the feasible region (Wang et al., 2008). Thus, this paper is only concerned with those individuals with fewer constraint violations in the population. In addition, it must be emphasized that the diversity in the population should be kept.

Motivated by these considerations, only the first half of the individuals are extracted from the combined population $(P_t + Q_t)$ after ranking the individuals based on their constraint violations in ascending order. The computational effort of the above process is $O((\mu + \lambda)\log(\mu + \lambda))$. As a result, a temporary population P_{temp} of size $(\mu + \lambda)/2$ is obtained. We select the first $\mu/2$ individuals from P_{temp} in order to steer the population toward feasibility; thereby, we expect to end with a feasible solution. Thereafter, the other $\mu/2$ individuals are randomly selected from the remaining $\lambda/2$ individuals of P_{temp} to promote diversity. These μ individuals form the next population. Figure 1 illustrates the above procedure. Since the μ individuals of the next population P_{t+1} are

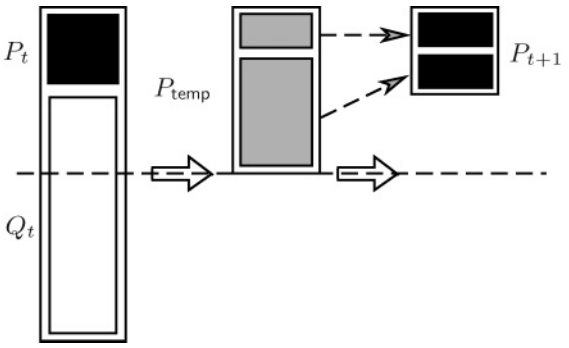

Figure 1: Schematic graph illustrating the constraint-handling mechanism in the first situation.

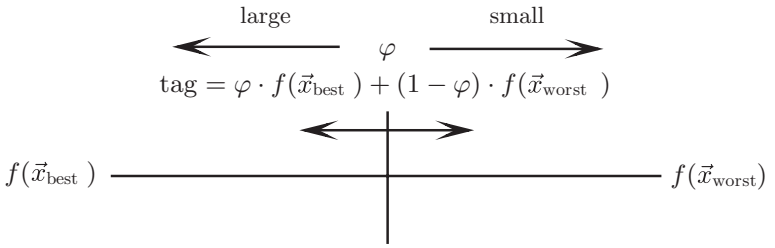

Figure 2: Graph representation for the constraint-handling mechanism in the second situation.

chosen from the individuals with fewer constraint violations in the combined population $(P_t + Q_t)$, the population will move constantly toward the feasible region during the evolution.

4.2.2 The Constraint-Handling Mechanism for the Semi-Feasible Situation

In this situation, it is expected that some important feasible individuals (feasible individuals with small objective function values) and infeasible individuals (infeasible individuals with slight constraint violations and small objective function values) should remain in the next population since such individuals are very promising for searching the optimal solution. An adaptive fitness transformation scheme is proposed to achieve this objective.

First, the combined population $(P_t + Q_t)$ is divided into the feasible group and the infeasible group according to the feasibility of each individual, and Z_1 and Z_2 are used to record the subscript of the individuals in the feasible and infeasible groups,

respectively:

$$Z_1 = \{i | G(\vec{x}_i) = 0, i = 1, \ldots, (\mu + \lambda)\} \tag{18}$$

$$Z_2 = \{i | G(\vec{x}_i) > 0, i = 1, \ldots, (\mu + \lambda)\}. \tag{19}$$

The best and worst feasible solutions (denoted as $\vec{x}_{\text{best}}$ and $\vec{x}_{\text{worst}}$) are found from the feasible group. Next, the objective function value $f(\vec{x}_i)((i \in \{1, \ldots, (\mu + \lambda)\})$ is converted into the following form (Wang et al., 2008):

$$f'(\vec{x}_i) = \begin{cases} f(\vec{x}_i), & i \in Z_1 \\ \max\{\varphi \cdot f(\vec{x}_{\text{best}}) + (1 - \varphi) \cdot f(\vec{x}_{\text{worst}}), f(\vec{x}_i)\}, & i \in Z_2 \end{cases} \tag{20}$$

where φ denotes the feasibility proportion of the last population $P_{t\text{-}1}$.

The above transformation can be explained by Figure 2. If the value of the parameter φ is large (meaning that the last population $P_{t\text{-}1}$ contains more feasible solutions than infeasible solutions), the tag will move left. As a result, the objective function values of infeasible individuals defined by Equation (20) are smaller; consequently, the probability of infeasible individuals surviving into the next population may increase. In contrast, if the value of the parameter φ is small (signifying that the last population $P_{t\text{-}1}$ consists of more infeasible solutions than feasible solutions), the tag will move right. Thus, the objective function values defined by Equation (20) for infeasible individuals are greater, which indicates feasible solutions may be selected with a higher probability. This adaptive transformation aims at maintaining a reasonable balance between feasible and infeasible solutions through observing the change of the feasibility proportion in the population over the course of evolution.

Each objective function value is then normalized:

$$f_{\text{nor}}(\vec{x}_i) = \frac{f'(\vec{x}_i) - \min\limits_{j \in Z_1 \cup Z_2} f'(\vec{x}_j)}{\max\limits_{j \in Z_1 \cup Z_2} f'(\vec{x}_j) - \min\limits_{j \in Z_1 \cup Z_2} f'(\vec{x}_j)}, \quad i \in \{1, \ldots, (\mu + \lambda)\}. \tag{21}$$

Afterward, the constraint violation should be normalized to have the same order of magnitude with the objective function value. Note that each constraint violation is normalized in the first criterion when computing the degree of constraint violation of each individual. With respect to the second criterion, the constraint violation is normalized as follows:

$$G_{\text{nor}}(\vec{x}_i) = \begin{cases} 0, & i \in Z_1 \\ \dfrac{G(\vec{x}_i) - \min\limits_{j \in Z_2} G(\vec{x}_j)}{\max\limits_{j \in Z_2} G(\vec{x}_j) - \min\limits_{j \in Z_2} G(\vec{x}_j)}, & i \in Z_2. \end{cases} \tag{22}$$

A final fitness function is obtained by adding the normalized objective function value and constraint violation together:

$$f_{\text{final}}(\vec{x}_i) = \begin{cases} f_{\text{nor}}(\vec{x}_i) + G(\vec{x}_i) & \text{if criterion} = 1 \\ f_{\text{nor}}(\vec{x}_i) + G_{\text{nor}}(\vec{x}_i) & \text{if criterion} = 2 \end{cases}, \quad i \in \{1, \ldots, (\mu + \lambda)\}. \tag{23}$$

In this situation, μ individuals with the smallest f_{final} are selected for the next population. By this adaptive fitness transformation scheme, some potential feasible and infeasible solutions may survive into the next generation.

4.2.3 The Constraint-Handling Mechanism for the Feasible Situation

In this situation, since the combined population $(P_t + Q_t)$ is composed of feasible solutions only, COPs can be considered as unconstrained optimization problems. Thus, the comparison of individuals is based only on the objective function values. In this situation, μ individuals with the smallest objective function values are selected and form the next population.

REMARK 1: With respect to the above three situations, since all previous and current population members are included in the combined population $(P_t + Q_t)$, elitism is ensured. In addition, the overall computational time complexity of these three situations is $O((\mu + \lambda)\log(\mu + \lambda))$.

REMARK 2: The two major differences between ATM and IATM are:

- In the infeasible situation, according to Jensen (2003), nondominated sorting can be done in $O((\mu + \lambda)\log(\mu + \lambda))$ for problems with two objectives, which means ATM and IATM have the same complexity under this condition; however, the implementation of IATM is much easier than that of ATM. More importantly, there is no need to compute the objective function values in the infeasible situation for IATM, which reduces the computational cost significantly compared with ATM, especially for problems where the evaluation of the objective function of a solution takes much time. Furthermore, based on the experiments in Section 6.4, IATM seems to be more effective than ATM for the $(\mu + \lambda)$-DE.
- Unlike the case for ATM, in IATM, two criteria are adopted to evaluate the degree of constraint violation of each individual, depending on the properties of the problems, which makes the performance of the algorithm more competitive.

4.3 Main Framework

The main framework of $(\mu + \lambda)$-CDE is shown in Algorithm 2. Initially, a population P_t of size μ is produced, and then population P_t is used to create an offspring population Q_t of size λ by the $(\mu + \lambda)$-DE introduced in Section 4.1. After integrating P_t with Q_t, IATM presented in Section 4.2 is used to select μ potential individuals for the next population. This procedure will continue until the termination criterion is satisfied.

5 Experimental Study

Numerical simulations were executed based on 24 well-known benchmark test functions (Liang et al., 2006) to evaluate the performance of the proposed $(\mu + \lambda)$-CDE. The details of this test suite are reported in Table 1, where $\rho = |\Omega|/|S|$ is the estimated ratio between the feasible region and the search space; LI is the number of linear inequality constraints; NI is the number of nonlinear inequality constraints; LE is the number of linear equality constraints; NE is the number of nonlinear equality constraints; a

t=0;// t denotes the generation number
generate an initial population $P_0 = \{\vec{x}_1, \ldots, \vec{x}_\mu\}$ by uniformly and randomly sampling from the search space S;
compute $\max_{j=1,\ldots,p} G_j^{\max} - \min_{j=1,\ldots,p} G_j^{\max}$, and determine which criterion should be used to compute the degree of constraint violation of each individual during the evolution;
compute the objective function value and the degree of constraint violation for each individual in the population P_0;
$\varphi = \frac{\text{num\_1}}{\mu}$, where num_1 denotes the number of feasible solutions in the population P_0;
repeat
  $t = t + 1$;
  $P_t = P_{t\text{-}1}$;
  $Q_t = \phi$;
  for *each individual in the population* P_t **do**
    execute the rand/1 strategy and the binomial crossover to generate the first trial vector $\vec{u}_{1,t}$;
    execute the improved current-to-best/1 strategy shown in Algorithm 1 to produce the second trial vector $\vec{u}_{2,t}$;
    execute the rand/2 strategy and the binomial crossover to create the third trial vector $\vec{u}_{3,t}$;
    Q_t=$Q_t \cup \vec{u}_{1,t} \cup \vec{u}_{2,t} \cup \vec{u}_{3,t}$;
  end
  compute the objective function value and the degree of constraint violation for each individual in the population Q_t;
  $fp = \frac{\text{num\_2}}{\mu+\lambda}$, where num_2 denotes the number of feasible solutions in the combined population $(P_t + Q_t)$;
  determinate the current situation of the combined population $(P_t + Q_t)$ according to the value of fp;
  P_t={the best μ individuals chosen from the combined population $(P_t + Q_t)$ based on the IATM};
  $\varphi = \frac{\text{num\_3}}{\mu}$, where num_3 denotes the number of feasible solutions in the population P_t;
until *the stopping criterion is met*;
output: the best individual of the population P_t

Algorithm 2: The framework of the proposed $(\mu + \lambda)$-CDE

is the number of constraints active at the optimal solution; and $f(\vec{x}^*)$ is the objective function value of the best known solution $\vec{x}^*$. Note that there are 11 test functions (i.e., test functions g03, g05, g11, g13, g14, g15, g17, g20, g21, g22, and g23) containing equality constraints. Since an improved best known solution has been found in this paper for test function g17: $\vec{x}^*$ = (201.784462493550, 99.999999999999, 383.071034852773, 419.999999999999, −10.907682614506, 0.073148231208) with $f(\vec{x}^*) = 8853.53387480648$, the $f(\vec{x}^*)$ of this test function in Table 1 is different from that in Liang et al. (2006).

Table 1: Details of 24 benchmark test functions.

Test function	n	Objective function	ρ	LI	NI	LE	NE	a	$f(\vec{x}^*)$
g01	13	quadratic	0.0111%	9	0	0	0	6	−15.0000000000
g02	20	nonlinear	99.9971%	0	2	0	0	1	−0.8036191042
g03	10	polynomial	0.0000%	0	0	0	1	1	−1.0005001000
g04	5	quadratic	51.1230%	0	6	0	0	2	−30665.5386717834
g05	4	cubic	0.0000%	2	0	0	3	3	5126.4967140071
g06	2	cubic	0.0066%	0	2	0	0	2	−6961.8138755802
g07	10	quadratic	0.0003%	3	5	0	0	6	24.3062090681
g08	2	nonlinear	0.8560%	0	2	0	0	0	−0.0958250415
g09	7	polynomial	0.5121%	0	4	0	0	2	680.6300573745
g10	8	linear	0.0010%	3	3	0	0	0	7049.2480205286
g11	2	quadratic	0.0000%	0	0	0	1	1	0.7499000000
g12	3	quadratic	4.7713%	0	1	0	0	0	−1.0000000000
g13	5	nonlinear	0.0000%	0	0	0	3	3	0.0539415140
g14	10	nonlinear	0.0000%	0	0	3	0	3	−47.7648884595
g15	3	quadratic	0.0000%	0	0	1	1	2	961.7150222899
g16	5	nonlinear	0.0204%	4	34	0	0	4	−1.9051552586
g17	6	nonlinear	0.0000%	0	0	0	4	4	8853.5338748065
g18	9	quadratic	0.0000%	0	13	0	0	0	−0.8660254038
g19	15	nonlinear	33.4761%	0	5	0	0	0	32.6555929502
g20	24	linear	0.0000%	0	6	2	12	16	0.2049794002
g21	7	linear	0.0000%	0	1	0	5	6	193.7245100700
g22	22	linear	0.0000%	0	1	8	11	19	236.4309755040
g23	9	linear	0.0000%	0	2	3	1	6	−400.0551000000
g24	2	linear	79.6556%	0	2	0	0	2	−5.5080132716

5.1 General Performance of the Proposed Approach for the 24 Benchmark Test Functions

For each test case, 25 independent runs were performed using the (70+210)-DE. F and CR were taken as 0.8 and 0.9, respectively. The mutation probability p_{m} was set to 0.05 and ξ was set to 200. The above parameters were set based on our experiments and were maintained in all runs. The maximum number of fitness evaluations (FEs) was equal to 500,000 as suggested by Liang et al. (2006). It is important to note that the parameter φ in Algorithm 1, Figure 2, Algorithm 2, and Equation (20) denotes the feasibility proportion of the last population $P_{t\text{-}1}$, so it does not need to adapt and can be obtained by computing the proportion of feasible solutions in the last population $P_{t\text{-}1}$.

Similar to Mezura-Montes and Coello (2005), we employed a dynamic setting of the parameter δ for equality constraints using the following equation:

$$\delta_{t+1} = \begin{cases} \frac{\delta_t}{\delta'} & \text{if } \delta_t > 0.0001 \\ 0.0001 & \text{otherwise} \end{cases} \tag{24}$$

where the initial δ_0 was set to $n \cdot \log 10(\max_{i=1,\ldots,n}(u_i - l_i))$ and the value of δ' was set to 1.015 based on our experiments. Note that the tolerance value for equality constraints is equal to 0.0001 when the algorithm halts so as to be in accord with Liang et al. (2006). A solution $\vec{x}$ is considered feasible in this paper only if $g_j(\vec{x}) \leq 0$, for $j = 1, \ldots, l$ and $|h_j(\vec{x})| - 0.0001 \leq 0$, for $j = l+1, \ldots, p$.

According to the suggestion in Liang et al. (2006), the function error value ($f(\vec{x}) - f(\vec{x}^*)$) of the achieved best solution $\vec{x}$ after 5×10^3, 5×10^4, and 5×10^5 FEs in each run is recorded in Tables 2–5. In these tables, SE indicates the standard error of the function error values obtained over 25 runs. Note that SE depends on both the standard deviation (SD) and the sample size, by the simple relation $\text{SE} = \text{SD}/\sqrt{\text{sample size}}$. The number of constraints that are violated at the median solution by more than 1.0, between 0.01 and 1.0, and between 0.0001 and 0.01, is shown in c, respectively. The values in the parentheses after the best, median, and worst function error values indicate the corresponding number of violated constraints.

Tables 2–5 reflect the detailed evolutionary behavior of our $(\mu + \lambda)$-CDE. As shown in Tables 2–5, for 12 test functions (i.e., test functions g01, g02, g04, g06, g07, g08, g09, g10, g12, g16, g19, and g24), the $(\mu + \lambda)$-CDE can enter the feasible region in each run by using 5×10^3 FEs. In addition, feasible solutions are found consistently for test functions g03 and g18 by using 5×10^4 FEs. With respect to the remaining 10 test functions (i.e., test functions g05, g11, g13, g14, g15, g17, g20, g21, g22, and g23), all of which contain equality constraints, $(\mu + \lambda)$-CDE finds feasible solutions in each run by making use of more FEs (i.e., 5×10^5 FEs) except for test functions g20 and g22. The above phenomenon occurs because we use the dynamic setting of the parameter δ for equality constraints, and the value of δ is able to reach 0.0001 after about 770 generations (i.e., 161,700 FEs) if $\delta_0 = 10$. As for test function g20, $(\mu + \lambda)$-CDE cannot find the feasible solutions in any run. However, it is noteworthy that test function g20 is a highly constrained problem (i.e., the feasibility proportion of the entire search space is extremely small), and no feasible solution has been reported so far. Concerning test function g22, $(\mu + \lambda)$-CDE can enter the feasible region several times.

The results reported in Tables 2–5 show that the $(\mu + \lambda)$-CDE has the capability of finding good feasible approximations of the best known solutions presented in Table 1 for test function g08 in 5×10^3 FEs and for test functions g04, g06, g09, g12, g16, and g24 in 5×10^4 FEs. By using 5×10^5 FEs, the resulting solutions derived from the $(\mu + \lambda)$-CDE are very close or even equal to the best known solutions except test functions g02, g21, and g22. It is worth noting that in terms of test functions g02 and g21, the $(\mu + \lambda)$-CDE can provide good feasible approximations of the best known solutions in the vast majority of runs. Since those test functions with equality constraints need relatively more FEs to reach the feasible region, the number of FEs acquired is 5×10^5 if they can provide good approximations of the best known solutions.

The number of FEs needed in each run to find a solution satisfying the following success condition: $f(\vec{x}) - f(\vec{x}^*) \leq 0.0001$ and $\vec{x}$ is feasible, is recorded in Table 6. For test function g20, when the population is very close to the feasible region in the later stage, the objective function value of the individual in the population is smaller than that of the best known solution; thus, the success condition is changed to $|f(\vec{x}) - f(\vec{x}^*)| \leq 0.0001$. In addition, the feasible rate (percentage of runs where at least one feasible solution is found), the success rate (percentage of runs where the algorithm finds a solution that satisfies the success condition), and the success performance (the mean FEs for successful runs multiplied by the number of total runs and divided by the number of successful runs) are also presented in Table 6.

Table 6 exhibits the overall performance of our $(\mu + \lambda)$-CDE. It can be seen from Table 6 that the feasible rate of 100% has been accomplished for all the test functions except for test functions g20 and g22. The feasible rate is 16% for test function g22. On the other hand, the $(\mu + \lambda)$-CDE finds at least one successful run for 23 test functions.

Table 2: Function error values achieved when FEs $= 5 \times 10^3$, FEs $= 5 \times 10^4$, and FEs $= 5 \times 10^5$ for test functions g01–g06.

FEs	Statistical measures for function error values	Test function					
		g01	g02	g03	g04	g05	g06
5×10^3	Best	4.36E+00 (0)	4.69E−01 (0)	6.62E−01 (0)	4.16E+01 (0)	−2.15E+01 (3)	1.09E+00 (0)
	Median	5.96E+00 (0)	5.26E−01 (0)	1.00E+00 (1)	8.02E+01 (0)	−5.55E+01 (3)	3.48E+00 (0)
	Worst	6.91E+00 (0)	5.58E−01 (0)	9.96E−01 (1)	1.29E+02 (0)	−6.85E+01 (3)	7.63E+00 (0)
	c	0, 0, 0	0, 0, 0	0, 0, 1	0, 0, 0	2, 1, 0	0, 0, 0
	Mean	5.86E+00	5.19E−01	9.77E−01	8.22E+01	−4.67E+01	3.69E+00
	SE	1.22E−01	5.00E−03	1.37E−02	5.03E+00	5.03E+00	3.80E−01
5×10^4	Best	4.99E−03 (0)	2.29E−01 (0)	1.85E−02 (0)	1.38E−09 (0)	−3.77E+00 (3)	3.37E−11 (0)
	Median	1.44E−02 (0)	2.92E−01 (0)	2.03E−01 (0)	8.56E−09 (0)	−4.65E+00 (3)	3.37E−11 (0)
	Worst	2.81E−02 (0)	3.50E−01 (0)	9.40E−01 (0)	5.46E−08 (0)	−4.90E+00 (3)	3.37E−11 (0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 3, 0	0, 0, 0
	Mean	1.57E−02	2.89E−01	3.29E−01	1.26E−08	−4.57E+00	3.37E−11
	SE	1.15E−03	6.01E−03	5.56E−02	2.39E−09	4.53E−02	2.64E−27
5×10^5	Best	0.00E+00 (0)	7.53E−11 (0)	−1.00E−11 (0)	7.64E−11 (0)	−1.82E−12 (0)	3.37E−11 (0)
	Median	0.00E+00 (0)	2.72E−10 (0)	−1.00E−11 (0)	7.64E−11 (0)	−1.82E−12 (0)	3.37E−11 (0)
	Worst	0.00E+00 (0)	8.96E−03 (0)	−1.00E−11 (0)	7.64E−11 (0)	−1.82E−12 (0)	3.37E−11 (0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	0.00E+00	3.58E−04	−1.00E−11	7.64E−11	−1.82E−12	3.37E−11
	SE	0.00E+00	3.58E−04	1.20E−16	7.43E−13	5.57E−13	0.00E+00

Table 3: Function error values achieved when FEs $= 5 \times 10^3$, FEs $= 5 \times 10^4$, and FEs $= 5 \times 10^5$ for test functions g07–g12.

FEs	Statistical measures for function error values	Test function					
		g07	g08	g09	g10	g11	g12
5×10^3	Best	3.88E+01 (0)	8.24E−11 (0)	2.67E+01 (0)	1.96E+03 (0)	−3.52E−01 (1)	8.83E−06 (0)
	Median	7.14E+01 (0)	9.95E−11 (0)	5.44E+01 (0)	2.84E+03 (0)	−3.91E−01 (1)	6.31E−05 (0)
	Worst	1.91E+02 (0)	3.45E−10 (0)	8.75E+01 (0)	4.86E+03 (0)	−4.18E−01 (1)	6.65E−03 (0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 1, 0	0, 0, 0
	Mean	8.07E+01	1.17E−10	5.66E+01	2.96E+03	−3.87E−01	3.75E−04
	SE	6.17E+00	1.08E−11	3.16E+00	1.44E+02	5.09E−03	2.63E−04
5×10^4	Best	7.38E−02 (0)	8.20E−11 (0)	5.92E−07 (0)	7.53E+00 (0)	−1.10E−03 (1)	0.00E+00 (0)
	Median	1.64E−01 (0)	8.20E−11 (0)	3.52E−06 (0)	1.33E+01 (0)	−1.66E−02 (1)	0.00E+00 (0)
	Worst	2.58E−01 (0)	8.20E−11 (0)	9.59E−06 (0)	3.18E+01 (0)	−1.72E−02 (1)	0.00E+00 (0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 1, 0	0, 0, 0
	Mean	1.59E−01	8.20E−11	4.42E−06	1.52E+01	−1.53E−02	0.00E+00
	SE	9.07E−03	2.83E−18	4.98E−07	1.30E+00	7.89E−04	0.00E+00
5×10^5	Best	7.98E−11 (0)	8.20E−11 (0)	−9.82E−11 (0)	6.28E−11 (0)	0.00E+00 (0)	0.00E+00 (0)
	Median	7.98E−11 (0)	8.20E−11 (0)	−9.81E−11 (0)	6.37E−11 (0)	0.00E+00 (0)	0.00E+00 (0)
	Worst	7.98E−11 (0)	8.20E−11 (0)	−9.81E−11 (0)	8.54E−08 (0)	0.00E+00 (0)	0.00E+00 (0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	7.98E−11	8.20E−11	−9.82E−11	9.83E−09	0.00E+00	0.00E+00
	SE	7.97E−16	2.83E−18	5.96E−14	4.16E−09	0.00E+00	0.00E+00

Table 4: Function error values achieved when FEs $= 5 \times 10^3$, FEs $= 5 \times 10^4$, and FEs $= 5 \times 10^5$ for test functions g13–g18.

FEs	Statistical measures for function error values	Test function					
		g13	g14	g15	g16	g17	g18
5×10^3	Best	1.32E−01 (3)	−1.92E+02 (3)	−2.28E−01 (2)	1.22E−02 (0)	−3.29E+02 (4)	2.64E−01 (1)
	Median	4.62E−01 (3)	−1.88E+02 (3)	−2.46E+00 (2)	4.15E−02 (0)	−4.08E+02 (4)	1.36E+00 (5)
	Worst	1.54E−01 (3)	−2.14E+02 (3)	−3.11E+00 (2)	7.67E−02 (0)	−5.32E+02 (4)	2.90E−01 (9)
	c	0, 3, 0	3, 0, 0	1, 1, 0	0, 0, 0	4, 0, 0	4, 1, 0
	Mean	1.59E−01	−2.00E+02	−2.01E+00	4.20E−02	−4.42E+02	4.94E−01
	SE	2.64E−02	1.96E+00	1.98E−01	2.91E−03	2.03E+01	9.27E−02
5×10^4	Best	−5.42E−04 (3)	−3.18E+00 (3)	−5.73E−03 (2)	1.41E−09 (0)	5.23E+00 (4)	1.45E−02 (0)
	Median	−6.81E−03 (3)	−3.84E+00 (3)	−7.39E−02 (2)	3.51E−09 (0)	−1.88E+01 (4)	2.37E−02 (0)
	Worst	−8.50E−03 (3)	−6.78E+00 (3)	−1.21E−01 (2)	1.59E−08 (0)	−2.65E+01 (4)	3.58E−02 (0)
	c	0, 2, 1	0, 3, 0	0, 2, 0	0, 0, 0	0, 4, 0	0, 0, 0
	Mean	−6.42E−03	−4.51E+00	−8.08E−02	4.52E−09	−1.84E+01	2.42E−02
	SE	4.02E−04	2.49E−01	8.77E−03	6.80E−10	1.77E+00	1.38E−03
5×10^5	Best	4.19E−11 (0)	8.51E−12 (0)	6.08E−11 (0)	6.52E−11 (0)	−1.82E−11 (0)	1.78E−11 (0)
	Median	4.19E−11 (0)	8.51E−12 (0)	6.08E−11 (0)	6.52E−11 (0)	−1.82E−11 (0)	3.85E−11 (0)
	Worst	4.19E−11 (0)	8.51E−12 (0)	6.08E−11 (0)	6.52E−11 (0)	−1.82E−11 (0)	8.95E−09 (0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	4.19E−11	8.51E−12	6.08E−11	6.52E−11	−1.82E−11	4.57E−10
	SE	3.41E−18	5.07E−15	1.16E−13	9.06E−17	2.03E−13	3.56E−10

Table 5: Function error values achieved when FEs $= 5 \times 10^3$, FEs $= 5 \times 10^4$, and FEs $= 5 \times 10^5$ for test functions g19–g24.

FEs	Statistical measures for function error values	Test function					
		g19	g20	g21	g22	g23	g24
5×10^3	Best	1.49E+02 (0)	2.80E+00 (20)	2.30E+02 (5)	1.77E+04 (19)	−1.42E+03 (4)	1.94E−05 (0)
	Median	2.72E+02 (0)	5.97E+00 (20)	−3.50E+01 (5)	1.86E+04 (19)	−1.44E+03 (4)	8.69E−04 (0)
	Worst	3.63E+02 (0)	8.74E+00 (20)	3.76E+01 (5)	1.09E+04 (19)	−1.39E+03 (4)	4.30E−04 (0)
	c	0, 0, 0	1, 17, 1	2, 2, 1	16, 3, 0	3, 1, 0	0, 0, 0
	Mean	2.73E+02	5.77E+00	2.13E+01	1.54E+04	−1.39E+03	1.15E−04
	SE	1.03E+01	2.94E−01	1.55E+01	8.68E+02	2.56E+01	1.72E−05
5×10^4	Best	2.21E+00 (0)	7.65E−02 (19)	−1.94E+02 (5)	9.07E+02 (19)	−2.35E+02 (4)	4.67E−12 (0)
	Median	3.35E+00 (0)	7.53E−02 (20)	−1.94E+02 (5)	1.95E+04 (19)	−2.45E+02 (4)	4.67E−12 (0)
	Worst	4.74E+00 (0)	1.07E−01 (19)	−1.90E+02 (5)	3.65E+03 (19)	−2.59E+02 (4)	4.67E−12 (0)
	c	0, 0, 0	0, 7, 13	0, 5, 0	16, 3, 0	0, 4, 0	0, 0, 0
	Mean	3.29E+00	1.39E−01	−1.93E+02	1.28E+04	−2.12E+02	4.67E−12
	SE	1.43E−01	1.68E−02	6.74E−01	1.31E+03	1.32E+01	1.81E−16
5×10^5	Best	1.41E−09 (0)	−4.71E−06 (7)	−6.31E−11 (0)	2.34E+00 (0)	6.84E−10 (0)	4.67E−12 (0)
	Median	3.39E−09 (0)	−4.72E−06 (8)	−9.65E−11 (0)	1.00E+04 (4)	2.57E−09 (0)	4.67E−12 (0)
	Worst	1.34E−08 (0)	−4.71E−06 (10)	1.31E+02 (0)	3.89E+03 (2)	8.18E−09 (0)	4.67E−12 (0)
	c	0, 0, 0	0, 1, 7	0, 0, 0	3, 0, 1	0, 0, 0	0, 0, 0
	Mean	4.42E−09	−4.72E−06	1.05E+01	1.26E+04	3.52E−09	4.67E−12
	SE	6.53E−10	1.69E−09	7.25E+00	1.43E+03	5.02E−10	1.81E−16

Table 6: Number of FEs to achieve the success condition, success rate, feasible rate, and success performance.

Test function	Best	Median	Worst	Mean	SE	Feasible rate	Success rate	Success performance
g01	80,920	89,320	97,720	89,000	893.7	100%	100%	89,000
g02	210,490	272,860	364,000	277,379	7,489.7	100%	96%	288,936
g03	71,470	107,170	184,660	111,025	5,474.4	100%	100%	111,025
g04	28,000	30,310	33,040	30,620	296.5	100%	100%	30,620
g05	164,080	165,130	165,550	165,079	78.3	100%	100%	165,079
g06	8,470	11,200	12,250	11,032	158.1	100%	100%	11,032
g07	130,690	139,720	153,580	141,038	1,177.3	100%	100%	141,038
g08	1,120	2,170	2,380	2,010	62.1	100%	100%	2,010
g09	36,190	39,550	46,060	39,953	466.7	100%	100%	39,953
g10	175,210	188,860	211,750	188,725	1,945.9	100%	100%	188,725
g11	58,240	73,360	119,560	79,475	3,214.4	100%	100%	79,475
g12	1,330	5,110	6,790	4,908	219.1	100%	100%	4,908
g13	141,820	148,960	149,590	148,237	380.9	100%	100%	148,237
g14	170,800	176,260	184,450	176,671	697.5	100%	100%	176,671
g15	111,580	127,750	145,390	130,622	2,276.8	100%	100%	130,622
g16	17,500	18,970	21,280	19,154	231.4	100%	100%	19,154
g17	178,780	183,820	188,230	183,962	535.3	100%	100%	183,962
g18	148,540	218,050	287,980	215,068	6,770.8	100%	100%	215,068
g19	234,220	265,930	291,970	268,374	3,178.6	100%	100%	268,374
g20	95,830	129,640	256,900	148,506	10,183.2	0%	100%	148,506
g21	199,570	210,700	220,150	209,896	1,151.1	100%	92%	228,148
g22	—	—	—	—	—	16%	0%	—
g23	241,990	263,830	281,260	263,695	2,096.2	100%	100%	263,695
g24	4,270	5,110	5,950	5,059	84.2	100%	100%	5,059

Table 7: Mean number of FEs to find the first feasible solution for 24 benchmark test functions in 25 runs.

Test function	Mean number of FEs	Test function	Mean number of FEs	Test function	Mean number of FEs	Test function	Mean number of FEs
g01	1,304	g07	1,901	g13	148,120	g19	70
g02	70	g08	70	g14	153,008	g20	—
g03	6,076	g09	313	g15	126,952	g21	164,323
g04	70	g10	1,682	g16	784	g22	350,875
g05	164,987	g11	58,240	g17	163,399	g23	169,380
g06	910	g12	70	g18	8,486	g24	70

The success rate is 100% for all the test functions with the exception of test functions g02, g21, and g22. For test functions g02 and g21, one or two runs of the $(\mu + \lambda)$-CDE cannot achieve the target accuracy level. As far as the success performance is concerned, $(\mu + \lambda)$-CDE requires fewer than 5,000 FEs for two test functions, fewer than 50,000 FEs for seven test functions, fewer than 200,000 FEs for 18 test functions, and fewer than 300,000 FEs for 23 test functions, to attain the success condition.

In order to further ascertain the velocity that $(\mu + \lambda)$-CDE enters the feasible region, the mean number of FEs to find the first feasible solution for 24 benchmark test functions over 25 runs is summarized in Table 7. As shown in Table 7, the $(\mu + \lambda)$-CDE can

Table 8: Comparison of the $(\mu + \lambda)$-CDE with respect to MPDE (Tasgetiren and Suganthan, 2006), GDE (Kukkonen and Lampinen, 2006), jDE-2 (Brest et al., 2006), and MDE (Mezura-Montes et al., 2006) in terms of feasible rate and success rate. The best result for each test function among the compared algorithms is highlighted in **boldface**.

	Feasible rate					Success rate				
Test function	MPDE	GDE	jDE-2	MDE	$(\mu + \lambda)$-CDE	MPDE	GDE	jDE-2	MDE	$(\mu + \lambda)$-CDE
g02	**100%**	**100%**	**100%**	**100%**	**100%**	92%	72%	92%	16%	**96%**
g03	**100%**	96%	**100%**	**100%**	**100%**	84%	4%	0%	**100%**	**100%**
g05	**100%**	96%	**100%**	**100%**	**100%**	**100%**	92%	68%	**100%**	**100%**
g11	**100%**	**100%**	**100%**	**100%**	**100%**	96%	**100%**	96%	**100%**	**100%**
g13	88%	88%	**100%**	**100%**	**100%**	48%	40%	0%	**100%**	**100%**
g14	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	96%	**100%**	**100%**	**100%**
g15	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	96%	96%	**100%**	**100%**
g17	96%	76%	**100%**	**100%**	**100%**	28%	16%	4%	**100%**	**100%**
g18	**100%**	84%	**100%**	**100%**	**100%**	**100%**	76%	**100%**	**100%**	**100%**
g19	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	88%	**100%**	0%	**100%**
g20	0%	0%	**4%**	0%	0%	0%	0%	0%	0%	**100%**
g21	**100%**	88%	**100%**	**100%**	**100%**	68%	60%	92%	**100%**	92%
g22	0%	0%	0%	0%	**16%**	0%	0%	0%	0%	0%
g23	**100%**	88%	**100%**	**100%**	**100%**	**100%**	40%	92%	**100%**	**100%**
Mean	91.0%	88.2%	91.8%	91.7%	**92.3%**	84.0%	74.2%	76.7%	84.0%	**95.3%**

find feasible solutions in the initial population for six test functions. In addition, the $(\mu + \lambda)$-CDE requires fewer than 10,000 FEs to find the first feasible solution for 14 test functions, which include 13 test functions with inequality constraints and one test function with an equality constraint (i.e., test function g03). Regarding test functions with equality constraints, except for test functions g03, g20, and g22, the number of FEs necessary for the $(\mu + \lambda)$-CDE to find the first feasible solution ranges from 50,000 to 170,000.

5.2 Comparison with Some Other State of the Art Methods in Constrained Evolutionary Optimization

The $(\mu + \lambda)$-CDE was compared in terms of two performance metrics (i.e., feasible rate and success rate) against four state of the art approaches proposed in the CEC2006 special session on constrained real-parameter optimization: MPDE (Tasgetiren and Suganthan, 2006), GDE (Kukkonen and Lampinen, 2006), jDE-2 (Brest et al., 2006), and MDE (Mezura-Montes et al., 2006). Note that DE was adopted by these five methods in comparison, and the number of runs provided by them was 25. Although εDE (Takahama and Sakai, 2006) and SaDE (Huang et al., 2006) were also proposed in the CEC2006 special session on constrained real-parameter optimization, both of them applied classical mathematical methods as the local search operator and need gradient information. Therefore, it is unreasonable to compare pure EA algorithms with these two methods; as a result, they are not included in our comparisons.

The performance comparisons are shown in Table 8. The experimental results for MPDE, GDE, jDE-2, and MDE are directly taken from Tasgetiren and Suganthan (2006), Kukkonen and Lampinen (2006), Brest et al. (2006), and Mezura-Montes et al. (2006),

respectively. Note that the experimental results of those test functions for which all the methods in the comparison group can achieve both 100% feasible rate and 100% success rate are omitted due to space limitations.

As shown in Table 8, the mean feasible rate of $(\mu + \lambda)$-CDE is superior to that of MPDE, GDE, jDE-2, and MDE. The corresponding number of test functions for which MPDE, GDE, jDE-2, MDE, and $(\mu + \lambda)$-CDE are unable to consistently find feasible solutions in all runs is 4, 9, 2, 2, and 2, respectively. It is noteworthy that according to our best knowledge, $(\mu + \lambda)$-CDE is the first algorithm based on a pure evolutionary method that can find feasible solutions for test function g22.

In terms of the mean success rate, $(\mu + \lambda)$-CDE is significantly better than the four competitors. The corresponding number of test functions for which MPDE, GDE, jDE-2, MDE, and $(\mu + \lambda)$-CDE cannot consistently solve in all runs is 8, 13, 11, 4, and 3, respectively. It can be seen from Table 8 that test functions g20 and g22 pose a major challenge to these five methods. However, $(\mu + \lambda)$-CDE is able to consistently attain the success condition in all runs for test function g20.

The above discussion indicates that the overall performance of $(\mu + \lambda)$-CDE, when measured by feasible rate and success rate, is better than that of the four competitors, which are the state of the art methods in constrained evolutionary optimization.

6 Discussion

The effectiveness of some mechanisms adopted by $(\mu + \lambda)$-CDE was verified using the 24 benchmark test functions. As a performance indicator, the average and SE of the function error values $(f(\vec{x}) - f(\vec{x}^*))$ of the achieved best solutions $\vec{x}$ when the iteration terminated in 25 runs were calculated. Afterward, a t-test was employed for the purpose of comparing pair-wise algorithms based on this performance indicator. If an algorithm cannot consistently find feasible solutions for a test function in all runs, the feasible rate, rather than the average and SE of the function error values, was given for this test function. In this section, the experimental results of those test functions for which an identical success rate can be achieved by the compared methods are omitted in order to save space and make the comparison compact.

6.1 Effectiveness of the Three Mutation Strategies

In order to verify the rationality of our selection for the three mutation strategies (rand/1, current-to-best/1, and rand/2), three additional experiments were performed. The first experiment is the $(\mu + \lambda)$-CDE with the best/1 strategy (denoted as $(\mu + \lambda)$-CDE with B/1); the second experiment is the $(\mu + \lambda)$-CDE with the best/2 strategy (denoted as $(\mu + \lambda)$-CDE with B/2); and the last experiment is the $(\mu + \lambda)$-CDE with both the best/1 and the best/2 strategies (denoted as $(\mu + \lambda)$-CDE with B/1 and B/2). In the above experiments, after the best/1 strategy or the best/2 strategy was implemented, binomial crossover was used to generate a trial vector. The results derived from the above three experiments were compared with respect to the results provided by the $(\mu + \lambda)$-CDE. Table 9 summarizes the experimental results for test functions g02, g18, g21, and g22.

As shown in Table 9, the $(\mu + \lambda)$-CDE is statistically better than the three competitors on two test functions respectively, according to the t-test. In addition, the $(\mu + \lambda)$-CDE with B/2 cannot outperform the $(\mu + \lambda)$-CDE on even one test function, and the

Table 9: Comparison of $(\mu + \lambda)$-CDE with respect to three variants on test functions g02, g18, g21, and g22 over 25 independent runs. Mean Value and SE indicate the average and standard error of the minimum function error values obtained in 25 runs, respectively. Results in parentheses denote the success rate. Results in square brackets denote the feasible rate. t-test is performed between $(\mu + \lambda)$-CDE and each of the three variants.

Problem	$(\mu + \lambda)$-CDE Mean value ± SE	$(\mu + \lambda)$-CDE with B/1 Mean value ± SE	$(\mu + \lambda)$-CDE with B/2 Mean value ± SE	$(\mu + \lambda)$-CDE with B/1 and B/2 Mean value ± SE
g02	3.58E–04 ± 3.58E–04 (96%)	2.30E–02 ± 4.94E–03* (24%)	3.58E–04 ± 3.58E–04 (96%)	1.58E–02 ± 4.10E–03* (36%)
g18	4.57E–10 ± 3.56E–10 (100%)	1.55E–11 ± 6.40E–18** (100%)	6.00E–05 ± 3.78E–05* (88%)	1.56E–11 ± 4.38E–14** (100%)
g21	1.05E+01 ± 7.25E+00 (92%)	2.61E+01 ± 1.07E+01* (80%)	2.61E+01 ± 1.07E+01* (80%)	4.71E+01 ± 1.28E+01* (64%)
g22	[16%] (0%)	[16%] (0%)	[4%] (0%)	[4%] (0%)

*The t value is significant at a 0.05 level of significance by two-tailed t-test. The corresponding algorithm is worse than $(\mu + \lambda)$-CDE.
**The t value is significant at a 0.05 level of significance by two-tailed t-test. The corresponding algorithm is better than $(\mu + \lambda)$-CDE.

$(\mu + \lambda)$-CDE with B/1 and the $(\mu + \lambda)$-CDE with B/1 and B/2 perform better than the $(\mu + \lambda)$-CDE on test function g18.

Test function g02 is known to have a very rugged fitness landscape and is in general very difficult to solve, for it includes 20 decision variables and a nonlinear objective function. In addition, it has a relatively large proportion of the feasible region. Therefore, this test function is mainly used to test the global search ability of the algorithm, and the diversity of the population should be maintained when solving this test function. Note that the $(\mu + \lambda)$-CDE with B/2 can obtain similar results with the $(\mu + \lambda)$-CDE for test function g02, which means adding the best/2 strategy to the $(\mu + \lambda)$-CDE does not result in a negative effect on the diversity of the population during the evolution. The main reason is that, in the best/2 strategy, two difference vectors are added to the base vector, which might lead to better perturbation than strategies with only one difference vector. In contrast, incorporating the best/1 strategy into the $(\mu + \lambda)$-CDE will induce obvious performance degradation for test function g02. Under this condition, the success rates are only 24% and 32% for the $(\mu + \lambda)$-CDE with B/1 and the $(\mu + \lambda)$-CDE with B/1 and B/2, respectively. This is because in the best/1 strategy, only one difference vector is added to the best individual of the population, and other individuals in the population promptly converge to the best individual.

With respect to test function g18 the $(\mu + \lambda)$-CDE, the $(\mu + \lambda)$-CDE with B/1, and the $(\mu + \lambda)$-CDE with B/1 and B/2 can achieve a 100% success rate; however, the $(\mu + \lambda)$-CDE with B/1 and the $(\mu + \lambda)$-CDE with B/1 and B/2 provide higher error accuracy than the $(\mu + \lambda)$-CDE. The above phenomenon signifies that the algorithms including the best/1 strategy might benefit from a faster convergence speed if all the algorithms in the comparison group can solve a specified test function. The error accuracy provided by the $(\mu + \lambda)$-CDE with B/2 for test function g18 is larger than the target error accuracy level (i.e., 0.0001) for three out of 25 runs. It means that combining the best/2 strategy with the $(\mu + \lambda)$-CDE might have a side effect on the convergence speed, which can be

further verified by test function g22. Test function g22 is a highly constrained problem that includes 22 decision variables, eight linear equality constraints, and 11 nonlinear equality constraints. For this test function, the feasible rates of the $(\mu + \lambda)$-CDE with B/2 and the $(\mu + \lambda)$-CDE with B/1 and B/2 are both 4%.

Test function g21 is also a highly constrained problem which contains seven decision variables and five nonlinear equality constraints. In order to reach the optimal solution of this test function, the capability of the algorithm to improve the feasible solutions after the population enters the feasible region is very important. As shown in Table 9, the success rates of the $(\mu + \lambda)$-CDE with B/1, the $(\mu + \lambda)$-CDE with B/2, and the $(\mu + \lambda)$-CDE with B/1 and B/2 are less than the success rate of the $(\mu + \lambda)$-CDE, which suggests that the $(\mu + \lambda)$-CDE provides higher performance for improving the previously found feasible solutions. It also suggests that the ability to improve the feasible solutions inside the feasible region might degrade if the $(\mu + \lambda)$-CDE frequently exploits the information of the best individual of the population due to the loss of diversity.

From the above discussion, we can conclude that the main drawback of combining the best/2 strategy with the $(\mu + \lambda)$-CDE is the deterioration of the convergence speed, although the diversity of the population can be kept under this condition. In addition, incorporating the best/1 strategy into the $(\mu + \lambda)$-CDE can improve the error accuracy for some test functions (e.g., g18); however, the information of the best solution may also cause premature convergence for some test functions.

In general, the average and SE of the number of FEs needed in each run to find a solution that satisfies the success condition can be considered as a performance indicator for assessing the convergence speed of an algorithm. Based on our observations, the difference of the convergence speed between the $(\mu + \lambda)$-CDE and the $(\mu + \lambda)$-CDE with B/2 was significant, which supports the above analysis of the $(\mu + \lambda)$-CDE with B/2. We summarize the experimental results of the $(\mu + \lambda)$-CDE and the $(\mu + \lambda)$-CDE with B/2 in Table 10 according to the above performance indicator. The t-test was used to compare these two algorithms.

From Table 10, we can see that the $(\mu + \lambda)$-CDE converges significantly faster than the $(\mu + \lambda)$-CDE with B/2 for nearly all the test functions, except that the $(\mu + \lambda)$-CDE is comparable to the $(\mu + \lambda)$-CDE with B/2 on test functions g04, g12, and g24 and is worse than the $(\mu + \lambda)$-CDE with B/2 on test function g01. Note that test functions g01, g04, g12, and g24 only have inequality constraints, and the type of objective function is quadratic or linear, so these four test functions are relatively easy to solve. The above phenomenon may be because the best/2 strategy is not capable of improving the quality of the solutions during the evolution for some test functions, compared with other mutation strategies. As a result, the $(\mu + \lambda)$-CDE with B/2 will consume μ FEs more than the $(\mu + \lambda)$-CDE at each generation (since at each generation the $(\mu + \lambda)$-CDE with B/2 uses four mutation strategies to generate mutant vectors while the $(\mu + \lambda)$-CDE uses three mutation strategies to generate mutant vectors), which leads to inefficiency.

The comparison above confirms that only selecting three mutation strategies is reasonable for our $(\mu + \lambda)$-CDE.

6.2 Effectiveness of the Improved Current-to-Best/1 Strategy

As described in Section 4.1, the current-to-best/1 strategy was improved to further enhance the global exploration capability of the $(\mu + \lambda)$-CDE. Another algorithm (denoted $(\mu + \lambda)$-CDE-1) was executed to verify the effectiveness of this improvement by still using the original current-to-best/1 strategy.

Table 10: Experimental results of $(\mu + \lambda)$-CDE and $(\mu + \lambda)$-CDE with B/2 over 25 independent runs. Mean FEs and SE indicate the average and standard error of the number of FEs required to achieve the success condition in 25 runs, respectively. t-test is performed between $(\mu + \lambda)$-CDE and $(\mu + \lambda)$-CDE with B/2.

Test function	$(\mu + \lambda)$-CDE Mean FEs ± SE	$(\mu + \lambda)$-CDE with B/2 Mean FEs ± SE	Test function	$(\mu + \lambda)$-CDE Mean FEs ± SE	$(\mu + \lambda)$-CDE with B/2 Mean FEs ± SE
g01	8.90E+04 ± 8.94E+02	8.04E+04 ± 7.60E+02**	g13	1.48E+05 ± 3.80E+02	1.96E+05 ± 4.84E+02*
g02	2.77E+05 ± 7.48E+03	3.40E+05 ± 8.76E+03*	g14	1.77E+05 ± 6.98E+02	2.34E+05 ± 7.90E+02*
g03	1.11E+05 ± 5.48E+03	1.49E+05 ± 6.14E+03*	g15	1.31E+05 ± 2.28E+03	1.70E+05 ± 3.32E+03*
g04	3.06E+04 ± 2.96E+02	3.05E+04 ± 3.18E+02	g16	1.92E+04 ± 2.32E+02	2.16E+04 ± 3.20E+02*
g05	1.65E+05 ± 7.84E+01	2.20E+05 ± 8.52E+01*	g17	1.84E+05 ± 5.36E+02	2.36E+05 ± 3.52E+02*
g06	1.10E+04 ± 1.58E+02	1.44E+04 ± 2.36E+02*	g18	2.15E+05 ± 6.78E+03	3.57E+05 ± 8.96E+03*
g07	1.41E+05 ± 1.18E+03	2.07E+05 ± 3.36E+03*	g19	2.68E+05 ± 3.18E+03	3.91E+05 ± 4.96E+03*
g08	2.01E+03 ± 6.22E+01	2.22E+03 ± 8.02E+01*	g20	1.49E+05 ± 1.02E+04	2.04E+05 ± 1.38E+04*
g09	4.00E+04 ± 4.66E+02	5.98E+04 ± 1.84E+03*	g21	2.10E+05 ± 1.15E+03	2.74E+05 ± 6.12E+03*
g10	1.89E+05 ± 1.95E+03	3.13E+05 ± 4.74E+03*	g22	—	—
g11	7.95E+04 ± 3.22E+03	9.45E+04 ± 2.82E+03*	g23	2.64E+05 ± 2.10E+03	3.43E+05 ± 3.56E+03*
g12	4.91E+03 ± 2.20E+02	5.31E+03 ± 2.08E+02	g24	5.06E+03 ± 8.42E+01	5.28E+03 ± 1.08E+02

*The t value is significant at a 0.05 level of significance by two-tailed t-test. The $(\mu + \lambda)$-CDE with B/2 is worse than the $(\mu + \lambda)$-CDE.
**The t value is significant at a 0.05 level of significance by two-tailed t-test. The $(\mu + \lambda)$-CDE with B/2 is better than the $(\mu + \lambda)$-CDE.

Table 11: Comparison of the $(\mu + \lambda)$-CDE with respect to the $(\mu + \lambda)$-CDE-1 on test functions g02, g21, and g22 over 25 independent runs. Mean Value and SE indicate the average and standard error of the minimum function error values obtained in 25 runs, respectively. The results in parentheses denote the success rate. The results in square brackets denote the feasible rate. The t-test is performed between the $(\mu + \lambda)$-CDE and the $(\mu + \lambda)$-CDE-1.

Test function	$(\mu + \lambda)$-CDE Mean value ± SE	$(\mu + \lambda)$-CDE-1 Mean value ± SE
g02	3.58E–04 ± 3.58E–04 (96%)	1.97E–02 ± 3.52E–03* (12%)
g21	1.05E+01 ± 7.25E+00 (92%)	5.34E+01 ± 1.28E+01* (56%)
g22	[16%] (0%)	[0%] (0%)

*The t value is significant at a 0.05 level of significance by two-tailed t-test. The $(\mu + \lambda)$-CDE-1 is worse than the $(\mu + \lambda)$-CDE.

Table 11 summarizes the experimental results for test functions g02, g21, and g22. As shown in Table 11, the $(\mu + \lambda)$-CDE is significantly better than the $(\mu + \lambda)$-CDE-1 on test functions g02 and g21 according to the t-test.

For test function g02, the feasibility proportion is 99.9971% as shown in Table 1; consequently, the global exploration capability of the algorithm plays a key role in the solution of this test function as analyzed previously. Considering the original current-to-best/1 strategy, the information of the best individual has been utilized over the whole evolution, which makes the algorithm less reliable and may lead to premature convergence due to the relatively greedy mutation strategy. In contrast, the improved current-to-best/1 strategy utilizes not only the information of the best solution but also the information of other individuals of the population after the population enters the feasible region. Thus, the improved current-to-best/1 strategy can maintain sufficiently high population diversity. As shown in Table 11, the success rate provided by the $(\mu + \lambda)$-CDE is 96% for this test function, compared with 12% of the $(\mu + \lambda)$-CDE-1.

Test function g21 is a highly constrained problem, so it is very difficult for the algorithm to find feasible solutions for this test function. Moreover, in order to find the feasible global optimum, the capability of the algorithm to move inside the feasible region is also vital after the first feasible solution has been found. Although the $(\mu + \lambda)$-CDE-1 can consistently reach the feasible region for this test function, the success rate is only 56%. This suggests that the first feasible solution found by the $(\mu + \lambda)$-CDE-1 is just slightly improved during the evolution for some trials. The experimental results in Table 11 indicate that the progress rate of the $(\mu + \lambda)$-CDE in the feasible region can be improved by exploiting the information of other individuals in the population and adding the improved BGA mutation.

For test function g22, the $(\mu + \lambda)$-CDE-1 fails to find feasible solutions in any trial. As stated previously, this test function poses a great challenge to the pure evolutionary methods in terms of finding feasible solutions. During the infeasible evolution stage, the main difference between the $(\mu + \lambda)$-CDE-1 and the $(\mu + \lambda)$-CDE is that in the latter, the improved BGA mutation is adopted, and the first scaling factor of the current-to-best/1 strategy is uniformly chosen between 0 and 1 as shown in Algorithm 1. It seems that

Table 12: Comparison of the $(\mu + \lambda)$-CDE with respect to the $(\mu + \lambda)$-CDE-2 and the $(\mu + \lambda)$-CDE-3 on test functions g20, g21, and g22 over 25 independent runs. Mean Value and SE indicate the average and standard error of the minimum function error values obtained in 25 runs, respectively. The results in parentheses denote the success rate. The results in square brackets denote the feasible rate. The t-test is performed between the $(\mu + \lambda)$-CDE and each of the $(\mu + \lambda)$-CDE-2 and the $(\mu + \lambda)$-CDE-3.

Problem	$(\mu + \lambda)$-CDE Mean value ± SE	$(\mu + \lambda)$-CDE-2 Mean value ± SE	$(\mu + \lambda)$-CDE-3 Mean value ± SE
g20	–4.71E–06 ± 1.69E–09 (100%)	5.33E–01 ± 2.78E–01* (0%)	–4.71E–06 ± 2.12E–09 (100%)
g21	1.05E+01 ± 7.25E+00 (92%)	1.05E+01 ± 7.25E+00 (92%)	1.10E+02 ± 9.90E+00* (16%)
g22	[16%] (0%)	[16%] (0%)	[0%] (0%)

*The t value is significant at a 0.05 level of significance by two-tailed t-test. The corresponding algorithm is worse than the $(\mu + \lambda)$-CDE.

such improvements are helpful for finding the first feasible solution for this test function since the $(\mu + \lambda)$-CDE can find feasible solutions several times. This is not difficult to understand because an algorithm with such improvements is capable of exploring more promising regions.

The above comparison not only demonstrates that the global search ability of the $(\mu + \lambda)$-CDE-1 is quite limited but also verifies the effectiveness of the improved current-to-best/1 strategy.

6.3 Effect of the Two Criteria to Compute the Constraint Violation in the Semi-Feasible Situation

Compared with ATM, two criteria are adopted by IATM to compute the degree of constraint violation of an individual, according to the characteristics of the problems. Based on our observations, when solving test functions g02, g05, g06, g07, g09, g10, g16, g17, g18, g19, g21, g22, and g23, the $(\mu + \lambda)$-CDE was coupled with the first criterion, and when solving the rest of the test functions the $(\mu + \lambda)$-CDE was coupled with the second criterion. In order to investigate the effect of these two criteria, optimization runs were conducted for the 24 benchmark test functions. Note that the $(\mu + \lambda)$-CDE with only the first criterion is denoted as $(\mu + \lambda)$-CDE-2, and the $(\mu + \lambda)$-CDE with only the second criterion is denoted as $(\mu + \lambda)$-CDE-3. Table 12 summarizes the experimental results for test functions g20, g21, and g22, which means that the remaining 21 test functions are not sensitive to the criterion used.

The performance of the $(\mu + \lambda)$-CDE is significantly superior (Table 12) to that of the $(\mu + \lambda)$-CDE-2 on test function g20. For this function, the $(\mu + \lambda)$-CDE-2 is unable to reach the target error accuracy level in even one run. It is because the difference of the constraints for this test function is not remarkable (in the initial population $\max_{j=1,\cdots,p} G_j^{\max} - \min_{j=1,\cdots,p} G_j^{\max}$ is smaller than 1.7×10^2). As a result, this difference should be emphasized, and the second criterion is more suitable for this test function.

The $(\mu + \lambda)$-CDE surpasses the $(\mu + \lambda)$-CDE-3 on test function g21 (Table 12). The success rate of the $(\mu + \lambda)$-CDE is 92%, compared with 16% provided by the

$(\mu + \lambda)$-CDE-3. Also, the $(\mu + \lambda)$-CDE-3 faces difficulty in finding feasible solutions for test function g22. The performance degradation of the $(\mu + \lambda)$-CDE-3 is because the difference of the constraints is very significant for test functions g21 and g22 (in the initial population $\max_{j=1,\ldots,p} G_j^{\max} - \min_{j=1,\ldots,p} G_j^{\max}$ is larger than 6×10^3 and 1×10^8 for test function g21 and test function g22, respectively), and it is more reasonable to treat each constraint equally.

This comparison shows the advantage of calculating the degree of constraint violation of an individual according to the problems' properties. It is necessary to note that in this paper we use $G_j^{\max}$ of the initial population to measure the difference of the constraints. If $\max_{j=1,\ldots,p} G_j^{\max} - \max_{j=1,\ldots,p} G_j^{\max} > \xi$, the first criterion will be used to compute the degree of constraint violation; otherwise, the second criterion will be adopted. In this paper, ξ is set to 200 during the evolution. We observe that the $(\mu + \lambda)$-CDE is not sensitive to this parameter, and it can be chosen from a large range, such as 200–4000.

6.4 The Benefit of IATM and $(\mu + \lambda)$-DE

As pointed out previously, the performance of COEAs is mainly dependent on two components: the constraint-handling technique and the search algorithm. In this paper, we adopt IATM as the constraint-handling technique and the $(\mu + \lambda)$-DE as the search algorithm. Note that in our previous paper (Wang et al., 2008), ATM and ES served as the constraint-handling technique and the search algorithm, respectively. In principle, the classic DE takes the scaled difference vector of two randomly chosen population vectors to perturb a base vector, and the above process is considered as the mutation operation. In contrast, the classic ES uses predefined probability distribution functions to perturb the vectors in the population. In ES, the step size in each dimension is controlled by the predefined probability distribution functions. However, with respect to DE, the step size in each dimension is adapted dynamically since the difference vector is formed by the difference between two vectors randomly chosen from the population (Takahama and Sakai, 2009). Compared with the classic DE, the $(\mu + \lambda)$-DE proposed in this paper utilizes three different mutation strategies to generate trial vectors and a selection scheme similar to the $(\mu + \lambda)$-ES.

In order to identify the benefit of the two components of COEAs, we combined the two constraint-handling techniques (i.e., ATM and IATM) with the two search algorithms (i.e., $(\mu + \lambda)$-DE and ES) arbitrarily, and as a result, we obtained the following four algorithms: the $(\mu + \lambda)$-DE with IATM (i.e., the algorithm proposed in this paper), the $(\mu + \lambda)$-DE with ATM, the ES with IATM, and the ES with ATM (i.e., the algorithm proposed by Wang et al., 2008). The performance of these four algorithms was tested by making use of the 24 benchmark test functions. In order to make the comparison fair, the maximum number of FEs was set to 240,000 which was suggested by Wang et al. (2008). Summarized in Table 13 are the experimental results of the above four algorithms.

The experimental results shown in Table 13 suggest that the overall performance of the $(\mu + \lambda)$-DE with IATM and the $(\mu + \lambda)$-DE with ATM is the best and second best for this test suite, although both of them suffer from permutation convergence and converge slowly within the limited number of FEs for some test functions. The relatively better performance of the $(\mu + \lambda)$-DE with IATM and the $(\mu + \lambda)$-DE with ATM means that as a search algorithm, the $(\mu + \lambda)$-DE is more powerful than the ES to solve COPs. As for the $(\mu + \lambda)$-DE with IATM and the $(\mu + \lambda)$-DE with ATM, the former reaches

Table 13: Experimental results of the $(\mu + \lambda)$-DE with IATM, the $(\mu + \lambda)$-DE with ATM, the ES with IATM, and the ES with ATM after 240,000 FEs. Mean Value and SE indicate the average and standard error of the minimum function error values obtained in 25 independent runs, respectively. The results in parentheses denote the success rate. The results in square brackets denote the feasible rate. The best result for each test function among the compared algorithms is highlighted in **boldface**.

Test function	$(\mu + \lambda)$-DE with IATM Mean value ± SE	$(\mu + \lambda)$-DE with ATM Mean value ± SE	ES with IATM Mean value ± SE	ES with ATM Mean value ± SE
g01	4.29E–13 ± 5.00E–14 (100%)	4.29E–13 ± 1.00E–13 (100%)	**5.32E–15 ± 3.64E–15 (100%)**	1.06E–14 ± 4.08E–14 (100%)
g02	**9.26E–03 ± 3.18E–03 (23%)**	1.53E–02 ± 4.46E–03 (0%)	1.31E–02 ± 2.36E–03 (0%)	1.34E–02 ± 2.64E–03 (0%)
g03	2.00E–13 ± 1.92E–13 (100%)	**1.77E–15 ± 2.64E–15 (100%)**	9.22E–05 ± 2.16E–05 (100%)	7.62E–05 ± 2.14E–05 (100%)
g04	9.09E–11 ± 2.22E–12 (100%)	9.09E–11 ± 2.22E–12 (100%)	9.09E–11 ± 2.22E–12 (100%)	9.09E–11 ± 2.22E–12 (100%)
g05	**2.72E–12 ± 1.85E–13 (100%)**	**2.72E–12 ± 1.85E–13 (100%)**	5.52E+01 ± 1.56E+01 (0%)	1.15E+00 ± 3.62E–01 (0%)
g06	3.37E–11 ± 3.70E–13 (100%)	3.37E–11 ± 3.70E–13 (100%)	3.37E–11 ± 3.70E–13 (100%)	3.37E–11 ± 3.70E–13 (100%)
g07	**1.06E–08 ± 1.56E–09 (100%)**	1.43E–08 ± 1.22E–09 (100%)	2.24E–01 ± 7.50E–03 (0%)	9.79E–03 ± 2.16E–03 (0%)
g08	8.19E–11 ± 5.64E–18 (100%)	8.19E–11 ± 5.76E–18 (100%)	8.19E–11 ± 1.15E–18 (100%)	8.19E–11 ± 1.45E–18 (100%)
g09	**9.84E–11 ± 8.90E–14 (100%)**	**9.84E–11 ± 8.90E–14 (100%)**	1.92E–03 ± 7.08E–04 (0%)	8.94E–03 ± 2.08E–03 (0%)
g10	**1.58E–06 ± 9.44E–07 (100%)**	3.95E–04 ± 8.48E–05 (27%)	2.25E+02 ± 3.08E+01 (0%)	2.01E+02 ± 2.36E+01 (0%)
g11	**0.00E+00 ± 0.00E+00 (100%)**	**0.00E+00 ± 0.00E+00 (100%)**	2.55E–04 ± 4.04E–05 (17%)	3.19E–04 ± 6.12E–05 (17%)

Table 13: Continued.

Test function	$(\mu+\lambda)$-DE with IATM Mean value ± SE	$(\mu+\lambda)$-DE with ATM Mean value ± SE	ES with IATM Mean value ± SE	ES with ATM Mean value ± SE
g12	**0.00E+00 ± 0.00E+00** (**100%**)	**0.00E+00 ± 0.00E+00** (**100%**)	1.87E–04 ± 2.04E–04 (100%)	1.87E–04 ± 2.04E–04 (100%)
g13	**4.18E–11 ± 4.56E–18** (**100%**)	**4.18E–11 ± 4.56E–18** (**100%**)	1.28E–05 ± 4.40E–06 (100%)	9.20E–06 ± 2.52E–06 (100%)
g14	**1.02E–09 ± 5.42E–10** (**100%**)	1.47E–09 ± 1.39E–10 (100%)	5.48E–02 ± 1.55E–02 (0%)	5.54E–02 ± 1.60E–02 (0%)
g15	**6.15E–11 ± 1.39E–13** (**100%**)	**6.15E–11 ± 1.39E–13** (**100%**)	2.84E–04 ± 3.86E–05 (0%)	2.51E–04 ± 3.48E–05 (0%)
g16	**6.52E–11 ± 1.35E–16** (**100%**)	**6.52E–11 ± 1.35E–16** (**100%**)	2.69E–03 ± 2.82E–04 (0%)	2.29E–03 ± 1.72E–04 (0%)
g17	**2.70E–07 ± 7.58E–08** (**100%**)	4.03E–07 ± 1.04E–07 (100%)	5.38E+01 ± 6.36E+00 (100%)	2.72E+01 ± 4.26E+00 (100%)
g18	**3.62E–05 ± 9.60E–06** (**90%**)	6.49E–05 ± 1.10E–05 (87%)	3.35E–02 ± 5.34E–03 (40%)	1.92E–02 ± 1.17E–01 (40%)
g19	**2.84E–04 ± 3.14E–05** (**0%**)	3.02E–04 ± 4.30E–05 (0%)	5.14E–01 ± 5.06E–02 (0%)	7.93E–01 ± 7.36E–02 (0%)
g20	**–6.07E–05 ± 8.61E–06** (**90%**)	–1.88E–04 ± 6.60E–05 (50%)	[0%] (0%)	[0%] (0%)
g21	**8.73E+00 ± 6.64E+00** (**93%**)	1.09E+02 ± 9.92E+00 (17%)	[0%] (0%)	[0%] (0%)
g22	[0%] (0%)	[0%] (0%)	[0%] (0%)	[0%] (0%)
g23	**6.70E–05 ± 6.43E–06** (**83%**)	8.95E–05 ± 6.32E–06 (80%)	[23%] (0%)	[0%] (0%)
g24	4.67E–12 ± 3.60E–16 (100%)	4.67E–12 ± 3.60E–16 (100%)	4.67E–12 ± 3.60E–16 (100%)	4.67E–12 ± 3.60E–16 (100%)

higher error accuracy for test functions g07, g14, and g17 for which both algorithms can obtain a success rate of 100% and achieves higher success rates for test functions g02, g10, g18, g20, g21, and g23. However, the latter has the capability to provide better error accuracy for only one test function, g03. Thus, we can conclude that IATM is more suitable for the $(\mu + \lambda)$-DE.

In addition, while the ES with IATM works very well for test function g01, its overall performance is poor since premature convergence frequently occurs for test functions g02, g05, g07, g09, g10, g11, g14, g15, g16, g18, and g19. Moreover, for test functions g21 and g22, no feasible solutions have been found. The ES with ATM is significantly better than the ES with IATM on test functions g05, g07, and g17 based on the t-test, but the performance of ES with ATM is also less satisfactory for some test functions. For example, for test functions g21, g22, and g23, the ES with ATM cannot find any feasible solutions. Note that for test function g23, the ES with IATM outperforms the ES with ATM since the feasible rate of the former is 23% compared with 0% of the latter. Moreover, the ES with IATM performs better than the ES with ATM on test functions g09 and g19 based on the t-test.

Although the above comparison indicates that the $(\mu + \lambda)$-DE exhibits a more powerful search ability than the ES, we cannot conclude that ATM is better than IATM or that IATM is better than ATM since they both show an overall similar performance when combined with ES according to the experimental results. The above comparison also indicates that certain search algorithms associated with specific constraint-handling techniques may be more effective than others. That is, the search algorithm and the constraint-handling technique should be combined in an effective manner in order to obtain competitive results. For example, the experimental results of Table 13 verify a mutually beneficial cooperation between IATM and the $(\mu + \lambda)$-DE.

6.5 About the Self-Adaptation of the $(\mu + \lambda)$-CDE

In this paper, we use three mutation strategies to generate the mutant vectors for the $(\mu + \lambda)$-CDE. It is of interest to determine whether employing a self-adaptive selection for the mutation strategies can improve the overall performance of our $(\mu + \lambda)$-CDE. Recently, Qin et al. (2009) proposed a DE with strategy adaptation, in which a mutation strategy candidate pool is maintained and the mutation strategies in the candidate pool are gradually self-adapted. The core idea of this approach is the following. For each target vector in the current population, one mutation strategy is selected from the candidate pool according to the probability learned from its success rate in generating improved solutions within a fixed number of previous generations (called the learning period, LP). Afterward, the selected mutation strategy is applied to generate a mutant vector. Based on the above idea of Qin et al. (2009), we presented a self-adaptive $(\mu + \lambda)$-CDE which is implemented as follows.

Suppose that the probability of selecting the kth mutation strategy is p_k, $k = 1, 2, \ldots, K$, where K is the number of mutation strategies in the candidate pool. Clearly, K is equal to three in this paper. In the initial LP generations, p_k is set to $1/K$ which means all strategies have the same probability of being selected. Then, roulette wheel selection is applied to choose the mutation strategy. In the $(\mu + \lambda)$-CDE, μ target vectors are exploited to produce λ trial vectors, and the best μ individuals are chosen from these μ target vectors and λ trial vectors as the potential individuals for the next generation. At generation t, we will record the number (denoted as $ns_{k,t}$) of the trial vectors generated by the kth mutation strategy which are allowed to survive to the next generation

and the number (denoted as $nf_{k,t}$) of the trial vectors generated by the kth mutation strategy which do not survive. Afterward, the above data will be stored in the success and failure memories as suggested by Qin et al. (2009). Once the number of generations is larger than LP, at each generation, the earliest records stored in the memories will be deleted so that the current data can be stored in the memories. After the initial LP generations, the probability of choosing different strategies will be updated at each generation based on the success and failure memories as follows:

$$S_{k,t} = \frac{\sum_{g=t-LP}^{t-1} ns_{k,g}}{\sum_{g=t-LP}^{t-1} ns_{k,g} + \sum_{g=t-LP}^{t-1} nf_{k,g}} + \varsigma, \quad (k = 1, 2, \ldots, K; t > LP) \tag{25}$$

and

$$p_{k,t} = \frac{S_{k,t}}{\sum_{k=1}^{K} S_{k,t}} \tag{26}$$

where $S_{k,t}$ denotes the success rate of the kth mutation strategy within the previous LP generations, and $\varsigma = 0.01$ is used to avoid $S_{k,t} = 0$. The larger $S_{k,t}$ means the kth mutation strategy will be selected with a larger probability to generate mutant vectors. Note that in our paper, three mutant vectors are produced for each target vector. Since roulette wheel selection is applied to choose the mutation strategy, the better mutation strategy (i.e., the mutation strategy with a larger $S_{k,t}$) will contribute more mutant vectors for each target vector at each generation.

The $(\mu + \lambda)$-CDE was compared with the self-adaptive $(\mu + \lambda)$-CDE using the 24 benchmark test functions over 25 runs. In our experiment, LP was set to 50 according to the suggestion in Qin et al. (2009). The performance of the self-adaptive $(\mu + \lambda)$-CDE has a significant difference with that of $(\mu + \lambda)$-CDE only on test function g22 based on our observations. Note that it is very difficult to find feasible solutions for this test function since this test function contains 19 equality constraints. According to the experimental results, the self-adaptive $(\mu + \lambda)$-CDE can further improve the feasible rate for this test function. The feasible rate provided by the self-adaptive $(\mu + \lambda)$-CDE is 40% for this test function, compared with 16% of the $(\mu + \lambda)$-CDE.

In addition, the convergence speed, which is measured by the average and SE of the number of FEs needed in each run for finding a solution satisfying the success condition, was compared between the $(\mu + \lambda)$-CDE and the self-adaptive $(\mu + \lambda)$-CDE. Table 14 records the experimental results. Based on the t-test, the self-adaptive $(\mu + \lambda)$-CDE converges significantly faster than the $(\mu + \lambda)$-CDE on 10 test functions and the $(\mu + \lambda)$-CDE exhibits faster convergence speed than the self-adaptive $(\mu + \lambda)$-CDE on one test function (g10). For the remaining test functions, the difference is marginal. Therefore, the self-adaptive $(\mu + \lambda)$-CDE shows overall better convergence speed than the $(\mu + \lambda)$-CDE on the 24 benchmark test functions.

According to the above comparison, we can draw the conclusion that the self-adaptive $(\mu + \lambda)$-CDE can further improve the performance of $(\mu + \lambda)$-CDE by enhancing the feasible rate and the convergence speed. The main reason is that in the self-adaptive $(\mu + \lambda)$-CDE, a more suitable mutation strategy can be determined self-adaptively to match different search phases by learning from the previous success experience. In addition, the percentage of each mutation strategy used in the self-adaptive $(\mu + \lambda)$-CDE during the evolution is summarized in Table 15.

Table 14: Experimental results of the $(\mu + \lambda)$-CDE and the self-adaptive $(\mu + \lambda)$-CDE over 25 independent runs. Mean FEs and SE indicate the average and standard error of the number of FEs required to achieve the success condition in 25 runs, respectively. The t-test is performed between the $(\mu + \lambda)$-CDE and the self-adaptive $(\mu + \lambda)$-CDE.

Test function	$(\mu + \lambda)$-CDE Mean FEs ± SE	Self-adaptive $(\mu + \lambda)$-CDE Mean FEs ± SE	Test function	$(\mu + \lambda)$-CDE Mean FEs ± SE	Self-adaptive $(\mu + \lambda)$-CDE Mean FEs ± SE
g01	8.90E+04 ± 8.94E+02	8.03E+04 ± 6.60E+02**	g13	1.48E+05 ± 3.80E+02	1.46E+05 ± 4.64E+02
g02	2.77E+05 ± 7.48E+03	2.11E+05 ± 5.62E+03**	g14	1.77E+05 ± 6.98E+02	1.69E+05 ± 5.82E+02**
g03	1.11E+05 ± 5.48E+03	8.09E+04 ± 3.60E+03**	g15	1.31E+05 ± 2.28E+03	1.30E+05 ± 2.32E+03
g04	3.06E+04 ± 2.96E+02	2.99E+04 ± 3.40E+02	g16	1.92E+04 ± 2.32E+02	1.91E+04 ± 2.56E+02
g05	1.65E+05 ± 7.84E+01	1.65E+05 ± 5.20E+01	g17	1.84E+05 ± 5.36E+02	1.81E+05 ± 9.96E+02
g06	1.10E+04 ± 1.58E+02	1.11E+04 ± 1.07E+02	g18	2.15E+05 ± 6.78E+03	1.58E+05 ± 4.96E+03**
g07	1.41E+05 ± 1.18E+03	9.71E+04 ± 9.08E+02**	g19	2.68E+05 ± 3.18E+03	1.91E+05 ± 2.04E+03**
g08	2.01E+03 ± 6.22E+01	2.11E+03 ± 8.08E+01	g20	1.49E+05 ± 1.02E+04	1.21E+05 ± 4.26E+03**
g09	4.00E+04 ± 4.66E+02	3.31E+04 ± 1.53E+02**	g21	2.10E+05 ± 1.15E+03	2.05E+05 ± 1.79E+03
g10	1.89E+05 ± 1.95E+03	2.92E+05 ± 4.95E+03*	g22	—	—
g11	7.95E+04 ± 3.22E+03	7.94E+04 ± 3.06E+03	g23	2.64E+05 ± 2.10E+03	2.27E+05 ± 1.30E+03**
g12	4.91E+03 ± 2.20E+02	4.93E+03 ± 1.88E+02	g24	5.06E+03 ± 8.42E+01	4.96E+03 ± 7.64E+01

*The t value is significant at a 0.05 level of significance by two-tailed t-test. The self-adaptive $(\mu + \lambda)$-CDE is worse than the $(\mu + \lambda)$-CDE.
**The t value is significant at a 0.05 level of significance by two-tailed t-test. The self-adaptive $(\mu + \lambda)$-CDE is better than the $(\mu + \lambda)$-CDE.

Table 15: The percentage of each mutation strategy used during the evolution for the 24 benchmark test functions.

Test function	The percentages of the rand/1 strategy, the improved current-to-best/1 strategy, and the rand/2 strategy	Test function	The percentages of the rand/1 strategy, the improved current-to-best/1 strategy, and the rand/2 strategy
g01	40%—42%—18%	g13	32%—42%—26%
g02	23%—63%—14%	g14	26%—57%—17%
g03	32%—45%—23%	g15	33%—38%—29%
g04	37%—40%—23%	g16	38%—38%—24%
g05	30%—44%—26%	g17	34%—45%—21%
g06	34%—34%—32%	g18	28%—56%—16%
g07	25%—59%—16%	g19	27%—57%—16%
g08	35%—32%—33%	g20	26%—62%—12%
g09	32%—43%—25%	g21	34%—42%—24%
g10	26%—59%—15%	g22	16%—73%—11%
g11	35%—36%—29%	g23	30%—52%—18%
g12	36%—38%—26%	g24	34%—32%—34%

7 Conclusion

We have proposed an efficient and effective COEA, named $(\mu + \lambda)$-CDE, to solve COPs, which combines the $(\mu + \lambda)$-DE with the improved adaptive trade-off model (IATM).

We investigated the performance of the proposed method using a benchmark suite consisting of 24 test functions. The experimental results suggested that the $(\mu + \lambda)$-CDE is very suitable for solving COPs with different types since it can achieve a 100% success rate for all of the test functions except for test functions g02, g21, and g22. For test functions g02 and g21, one or two runs are trapped in a local minimum. In addition, according to our best knowledge, the $(\mu + \lambda)$-CDE is the first pure evolutionary method to find feasible solutions for test function g22. Compared with some state of the art methods the $(\mu + \lambda)$-CDE is very competitive in terms of success rate and feasible rate. The mean success rate provided by the $(\mu + \lambda)$-CDE is 95.3%, which is significantly better than that of the four methods in the comparison group.

In addition, the rationale for using three mutation strategies (i.e., rand/1 strategy, current-to-best/1 strategy, and rand/2 strategy) was verified empirically. The effectiveness of the improved current-to-best/1 strategy and the two criteria to calculate the degree of constraint violation was studied. Moreover, we arbitrarily combined ES and $(\mu + \lambda)$-DE with ATM and IATM to see the benefit of the constraint-handling technique and the search algorithm. Finally, the $(\mu + \lambda)$-CDE was further improved by making use of the self-adaptive scheme introduced by Qin et al. (2009) and a self-adaptive $(\mu + \lambda)$-CDE was proposed.

In future studies, we will apply the proposed method to solve some real-world optimization problems.

The MATLAB source codes of the $(\mu + \lambda)$-CDE and the self-adaptive $(\mu + \lambda)$-CDE can be obtained from the first author upon request.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 60805027 and Grant 90820302, in part by the Research Fund for the Doctoral Program of Higher Education of China under Grant 200805330005, and in part by the Graduate Innovation Fund of Hunan Province of China under Grant CX2009B039. The authors sincerely thank the anonymous associate editor and the anonymous reviewers for their very helpful comments and suggestions which greatly improved the quality of this paper.

References

Becerra, R. L., and Coello, C. A. C. (2006). Cultured differential evolution for constrained optimization. *Computer Methods in Applied Mechanics and Engineering*, 195(33–36):4303–4322.

Brest, J. (2009). *Constrained real-parameter optimization with ε-self-adaptive differential evolution constraint-handling*. In *Studies in Computational Intelligence*, vol. 198, Chap. 3 (pp. 73–93). Berlin: Springer-Verlag.

Brest, J., Zumer, V., and Maucec, M. S. (2006). Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In *Proceedings of the Congress on Evolutionary Computation (CEC'2006)*, pp. 215–222.

Cai, Z., and Wang, Y. (2006). A multiobjective optimization-based evolutionary algorithm for constrained optimization. *IEEE Transactions on Evolutionary Computation*, 10(6):658–675.

Coello, C. A. C. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11–12):1245–1287.

Coit, D. W., Smith, A. E., and Tate, D. M. (1996). Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS Journal on Computing*, 8(2):173–182.

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2–4):311–338.

Huang, F., Wang, L., and He, Q. (2007). An effective co-evolutionary differential evolution for constrained optimization. *Applied Mathematics and Computation*, 186(1):340–356.

Huang, V. L., Qin, A. K., and Suganthan, P. N. (2006). Self-adaptive differential evolution algorithm for constrained real-parameter optimization. In *Proceedings of the Congress on Evolutionary Computation (CEC'2006)*, pp. 17–24.

Jensen, M. T. (2003). Reducing the run-time complexity of multiobjective EAs: The NSGA-ii and other algorithms. *IEEE Transactions on Evolutionary Computation*, 7(5):503–515.

Kukkonen, S., and Lampinen, J. (2006). Constrained real-parameter optimization with generalized differential evolution. In *Proceedings of the Congress on Evolutionary Computation (CEC'2006)*, pp. 207–214.

Lampinen, J. (2002). A constraint handling approach for the differential evolution algorithm. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, pp. 1468–1473.

Lampinen, J., and Zelinka, I. (1999a). Mixed integer-discrete-continuous optimization by differential evolution, Part 1: The optimization method. In *Proceedings of MENDEL'99, 5th International Mendel Conference on Soft Computing*, pp. 71–76.

Lampinen, J., and Zelinka, I. (1999b). Mixed integer-discrete-continuous optimization by differential evolution, Part 2: A practical example. In *Proceedings of MENDEL'99, 5th International Mendel Conference on Soft Computing*, pp. 77–81.

Lampinen, J., and Zelinka, I. (1999c). Mixed variable non-linear optimization by differential evolution. In *Proceedings of Nostradamus'99, 2nd International Prediction Conference*, pp. 45–55.

Liang, J. J., Runarsson, T. P., Mezura-Montes, E., Clerc, M., Suganthan, P. N., Coello, C. A. C., and Deb, K. (2006). Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. Technical report, Nanyang Technological University, Singapore.

Lin, Y. C., Hwang, K. S., and Wang, F. S. (2002). Hybrid differential evolution with multiplier updating method for nonlinear constrained optimization problems. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, pp. 872–877.

Mallipeddi, R., and Suganthan, P. N. (2010). Ensemble of constraint handling techniques. *IEEE Transactions on Evolutionary Computation*, doi: 10.1109/TEVC.2009.2033582.

Mezura-Montes, E. (Ed.). (2009). *Constraint-handling in evolutionary optimization*. Berlin: Springer.

Mezura-Montes, E., and Coello, C. A. C. (2005). A simple multimembered evolution strategy to solve constrained optimization problems. *IEEE Transactions on Evolutionary Computation*, 9(1):1–17.

Mezura-Montes, E., and Palomeque-Ortiz, A. G. (2009). Self-adaptive and deterministic parameter control in differential evolution for constrained optimization. In *Studies in Computational Intelligence*, Vol. 198, Chap. 3 (pp. 94–120). Berlin: Springer-Verlag.

Mezura-Montes, E., Velázquez-Reyes, J., and Coello, C. A. C. (2005). Promising infeasibility and multiple offspring incorporated to differential evolution for constrained optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2005)*, Vol. 1, pp. 225–232.

Mezura-Montes, E., Velázquez-Reyes, J., and Coello, C. A. C. (2006). Modified differential evolution for constrained optimization. In *Proceedings of the Congress on Evolutionary Computation (CEC'2006)*, pp. 332–339.

Michalewicz, Z., and Schoenauer, M. (1996). Evolutionary algorithm for constrained parameter optimization problems. *Evolutionary Computation*, 4(1):1–32.

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

Runarsson, T. P., and Yao, X. (2005). Search biases in constrained evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews*, 35(2):233–243.

Storn, R. (1999). System design by constraint adaptation and differential evolution. *IEEE Transactions on Evolutionary Computation*, 3(1):22–34.

Storn, R., and Price, K. (1995). Differential evolution—A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Science Institute, Berkeley.

Takahama, T., and Sakai, S. (2006). Constrained optimization by the ε-constrained differential evolution with gradient-based mutation and feasible elites. In *Proceedings of the Congress on Evolutionary Computation (CEC'2006)*, pp. 1–8.

Takahama, T., and Sakai, S. (2009). Solving difficult constrained optimization problems by the ε-constrained differential evolution with gradient-based mutation. In *Studies in Computational Intelligence*, Vol. 198, Chap. 3 (pp. 51–72). Berlin: Springer-Verlag.

Tasgetiren, M. F., and Suganthan, P. N. (2006). A multi-populated differential evolution algorithm for solving constrained optimization problem. In *Proceedings of the Congress on Evolutionary Computation (CEC'2006)*, pp. 33–40.

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

Venkatraman, S., and Yen, G. G. (2005). A generic framework for constrained optimization using genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 9(4):424–435.

Wang, Y., Cai, Z., Guo, G., and Zhou, Y. (2007). Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems. *IEEE Transactions on Systems Man and Cybernetics—Part B: Cybernetics*, 37(3):560–575.

Wang, Y., Cai, Z., Zhou, Y., and Zeng, W. (2008). An adaptive trade-off model for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 12(1):80–92.