Accepted by *Memetic Computing* on 5 March 2021. https://doi.org/10.1007/s12293-021-00335-8

An efficient bi-level differential evolution algorithm with adaptation of lower level population size and search radius

Lianghong Wu[1,*], Zhenzu Liu[1], Hua-Liang Wei[2], Rui Wang[3,*]

1. School of Information and Electrical Engineering, Hunan University of Science and Technology, Hunan Xiangtan, 411201, China
2. Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield S1 3JD, UK
3. School of Systems Engineering, National University of Defense Technology, Hunan Changsha, 410073, China
*Corresponding authors: lhwu@hnust.edu.cn, ruiwangnudt@gmail.com

Abstract: Bilevel optimization has been recognized as one of the most difficult and challenging tasks to deal with because a solution to the upper level problem may be feasible only if it is also an optimal solution to the lower level problem. In recent years, evolutionary bilevel optimization has attracted increasing interest. In this paper, an efficient self-adaptive bilevel differential evolution (SABiLDE) with *k*-nearest neighbors (*k*-NN) based interpolation is proposed to solve bilevel optimization problems. The *k*-NN approximation is applied to estimate the optimal lower level variables for any newly generated upper candidates to improve the computational efficiency. A similarity based self-adaptive strategy for the dynamic control of lower level population size and search radius is introduced to further enhance the efficiency of the lower level function evaluations. A test set with 10 standard test problems and the SMD suite with controllable complexities are used to evaluate the performance of the proposed approach. Compared with two recent state-of-the-art methods, the numerical results produced by the proposed method are promising and show great potential for solving generic bilevel optimization problems.

Keywords: Bilevel optimization; Differential evolution; *k*-Nearest neighbors learning; Self-adaptive strategy

1. Introduction

Bilevel optimization problems (BLOPs) are a special class of optimization problems, in which the lower level optimization problem acts as a constraint to the upper level problem and is required to solve first to get a feasible solution for the upper level optimization problem. Bilevel optimization problems widely exist in practice where a hierarchical decision-making is often required, particularly in economics [1]-[3], management [4], [5], transportation [6], [7], military [8], engineering [9], and others [10]-[11]. In comparison with traditional optimization problems, bilevel optimization problems are very difficult to solve as the nested structure of such a problem requires that a solution to the upper level problem should also be an optimal solution to the lower level problem. Mathematically, bilevel optimization is generally called bilevel programming. While in the domain of game theory, it is referred to as Stackelberg problem. A generic bilevel optimization problem can be formulated as follows [11]:

$$\underset{\boldsymbol{x}^u \in \Omega^u, \boldsymbol{x}^l \in \Omega^l}{\text{minimize}} \quad F(\boldsymbol{x}^u, \boldsymbol{x}^l)$$

$$\begin{aligned} \text{subject to} \quad & \boldsymbol{x}^l \in \operatorname{argmin}\{f(\boldsymbol{x}^u, \boldsymbol{x}^l) | \\ & \quad g_i(\boldsymbol{x}^u, \boldsymbol{x}^l) \leq 0, i = 1, \cdots, q^l \\ & \quad h_i(\boldsymbol{x}^u, \boldsymbol{x}^l) = 0, i = q^l + 1, \cdots, m^l\} \\ & G_j(\boldsymbol{x}^u, \boldsymbol{x}^l) \leq 0, j = 1, \cdots, q^u \\ & H_j(\boldsymbol{x}^u, \boldsymbol{x}^l) = 0, j = q^u + 1, \cdots, m^u \end{aligned} \tag{1}$$

where F and f are the objective functions at the upper and lower level, respectively. $\boldsymbol{x}^u$ represents the upper level decision vector and $\boldsymbol{x}^l$ represents the lower level decision vector. G_n and H_m, g_i and h_j are inequality and equality constraints at the upper and lower levels, respectively. Ω^u and Ω^l are the search spaces for the upper and lower level decision vectors. Note that the lower level optimization problem is optimized only with respect to the variables $\boldsymbol{x}^l$ and the variable vector $\boldsymbol{x}^u$ is kept fixed. However, the upper level optimization involves both variable vectors $\boldsymbol{x}^u$ and $\boldsymbol{x}^l$. The bilevel optimization problem generally needs to find the lower level optimal solution first and then search for the optimal solution for the upper level optimization problem.

In recent years, bilevel optimization has attracted much attention. However, due to the hierarchical structure in nature, solving a bilevel optimization problem is quite difficult and computationally demanding [11], [12]. It has been proved that even the simplest bilevel linear programming is strongly NP-hard [13], not to mention problems with complex nonlinear upper and lower objective functions and constraints. For example, if the lower level problem is a multimodal problem, there would be no guarantee that a lower level solution is the best for the upper level, and this can only lead to sub-optimal solutions for the upper level problem. Therefore, the development of an efficient bilevel optimization algorithm is a challenging task. The early work for bilevel optimization was mainly focused on classical methods under various assumptions, including Karush-Kuhn-Tucker (KKT) approach, branch-and-bound techniques, cutting plane algorithms, descent methods, penalty functions based methods, and so on [10], [14], [15]. However, most of the classical approaches can only be used to handle simple bilevel problems with good properties such as smoothness, linearity, quadratic or convexity [11]. To overcome the limitations of classical optimization methods, it is quite natural to introduce evolutionary algorithms (EAs) to solve bilevel optimization problems with higher levels of complexity, since EAs have a number of good algorithmic features such as derivative free, flexible and robust. However, it should be stressed that most of the existing bilevel evolutionary optimization algorithms are nested and involve intensive computational expense; the development of evolutionary algorithms for bilevel problems is still in an early stage. Thus, to significantly improve the performance of the existing approaches and develop new methods are still highly demanded [11], [12].

As a simple but efficient and versatile global optimization method, the Differential Evolution (DE) [16] algorithm has been used to solve bilevel optimization problems in recent years. For example, in order to reduce the number of upper and lower level function evaluations, a differential evolution method assisted by a k-nearest neighbors (k-NN) approach, namely BlDE, is proposed to solve bilevel programming problems using a surrogate model [17]. The surrogate model is chosen to replace the lower level DE algorithm in a specific probability. However, the approximation ability of the k-NN based surrogate model is limited due to the drawback that the surrogate model may produce an inaccurate lower level solution which can therefore lead to a false upper level solution; this is particular true for problems with conflict upper and lower tasks [18], [19]. Although this surrogate modeling approach is computationally efficient, the computation efficiency is gained by sacrificing the reliability and accuracy of the solutions.

To improve the efficiency of the k-NN approximation approach and meanwhile overcome its disadvantage, a novel self-adaptive bilevel differential evolution with k-NN approximation (SABiLDE) is proposed in this paper. The two major contributions of this paper are as follows: 1) *k*-nearest neighbors in the upper archive are identified for a newly generated upper candidate to better approximate the optimal lower level variables through the inverse distance weighting interpolating. If the corresponding upper level vectors are not proximal enough to the upper level vectors in the extern archive, the estimated lower level variables are marked to be a non-optimal solution and is not directly passed to the upper level task, but only used as a basic individual for the initialization of the lower level population. While a newly generated upper level vector is very similar to one of the archived members (the nearest neighbor distance is small enough), the approximated lower level solution based on *k*-NN interpolation would be accurate enough to replace the corresponding optimal lower level solution, and there is no need to invoke a lower level DE algorithm or another evolutionary algorithm. 2) based on the similarity (the nearest neighbor distance), a self-adaptive control rate is proposed to dynamically adjust the lower level population size and search radius to reduce the computation of the lower level function evaluations and therefore to improve the computational efficiency. Two different types of test problems are used to test the effectiveness of the proposed SABiLDE. Numerical results show that the SABiLDE has better computational efficiency and solution accuracy than the improved BlDE in [17]. In comparison with other four representative algorithms, the proposed SABiLDE has better or competitive accuracy and robustness for most of the test problems considered, and has better computational efficiency for high dimensional problems.

The rest of this paper is organized as follows. In Section 2, a literature review on evolutionary bilevel optimization is provided. Thereafter, the detailed descriptions of the framework of the SABiLDE algorithm are presented in Section 3. Numerical experiments and comparisons are illustrated in Section 4. Finally, Section 5 gives the concluding remarks.

2. Current research on evolutionary bilevel optimization

Bilevel optimization has been recognized as one of the most difficult and challenging tasks to deal with because of its intrinsic complexity in general. In the past decades, bilevel optimization has gained increasing interest due to its wide applications, and a number of classical methods and evolutionary algorithms have been developed to solve bilevel optimization problems. The traditional concepts and approaches for solving bilevel optimization problems were well reviewed in [10] and [13]. Interested readers are referred to these good surveys and the references therein. There has also been an interest in multi-objective bilevel optimization using evolutionary algorithms [7], [12], [20], [21]. Generally, the bilevel evolutionary algorithms (BLEAs) can be categorized into four groups: 1) single level transformation methods, 2) nested sequential strategies, 3) co-evolutionary approaches, and 4) surrogate model assisted methods. In the follows, a number of representative methods are briefly introduced.

2.1 Single level transformation methods

In these methods, a bilevel optimization problem is transformed into an equivalent single level optimization problem first, and an EA is then used to solve the equivalent problem. For example, Hejazi et al. [22] proposed a GA based method for linear bilevel programming, in which the Kuhn-Tucker conditions for the lower level problem are derived and then the bilevel programming problem is transferred into a single level problem. Wan et al. proposed a hybrid particle swarm optimization and chaos searching approach [23] and a distribution estimation algorithm [24] for solving bilevel programming problems. Similarly, the bilevel programming is transformed into a single level programming problem using the Karush-Kuhn-Tucker (KKT) conditions of the lower level problem. Wang et al. [25] proposed an EA for solving nonlinear bilevel programming problem (BLPP) based on a new constraint-handling scheme, where the nonlinear BLPP was firstly transferred into an equivalent nonlinear optimization problem with a single non-differentiable and nonconvex objective function. Moreover, a new constraint-handling method with linear and nonlinear schemes were combined with the EA [Ref]. In [26], Jiang et al. presented a novel approach based on particle swarm optimization to solve nonlinear bilevel programming problem, by applying the KKT conditions and the Chen-Harker-Kanzow-Smale (CHKS) smoothing function to the lower level problem first and then transforming the nonlinear BLPP into a regular nonlinear programming with complementary constraints. Recently, Li [27] proposed a GA approach using finite search space (GA-FSS) for a special class of BLPPs in which the lower level is a fractional program, whereas the upper level problem is simply solvable. In order to evaluate the performance of each individual in GA, a fitness function was presented by making use of the optimality conditions of linear fractional programs. Hence, the GA-FSS belongs to the single level transformation method in essence. The main disadvantage of the single level transformation method is that it is problem-dependent and lacks generalization ability to extend to new problems. To convert the lower level problem into complementary constraints of the upper level, the lower level problem must satisfy some special conditions, such as linear, convex and differentiable.

2.2 Nested strategies

Since the upper and lower tasks of bilevel optimization problems are nested in nature, it is reasonable to use a nested strategy to handle bilevel problems; one of the best ways is that for every upper level vector, to find a solution for the lower level optimization problem. There are two nested frameworks, depending on how the lower level problem is solved by using either a classical optimization method or an EA. It should be stressed that an EA is always adopted to solve the upper level task. One of the first nested strategies for handling bilevel optimization problems was proposed by Mathieu et al. [28], where the upper level problem was solved using a genetic algorithm (GA), while the lower level problem was handled by a linear programming method. Later, Yin [29] proposed a similar strategy based on GA, using the Frank Wolf algorithm to solve the lower level problem. In [30], the DE algorithm was combined with an interior point method for solving nonlinear problems with linear constraints. The DE algorithm was employed to solve the upper level problem while the interior point method was used to optimize the lower level problem. The DE was also used to solve a bilevel programming problem in transportation [31], where an optimal solution for the lower level problem was obtained via a gradient based algorithm. Islam et al [32] presented a new nested approach for solving bilevel optimization problems, which uses DE and a memetic algorithm adapted from the Sequential Quadratic Programming (SQP) to find a solution for the upper level model, and the DE and SQP are also used in the lower level during various phases of the search. The common characteristic of the above methods is that they all use a deterministic local search algorithm to handle the lower level problem. Similar to the single level transformation methods, the nested approaches with classical lower level methods are only applicable to problems with good lower level properties.

To overcome the disadvantages of the classical optimization algorithms and improve the generalization property of the existing bilevel optimization algorithms, EAs have also been proposed to solve the lower level problems. That is to say, both the upper and lower level tasks are optimized by EAs in a nested structure. For example, Zhang et al [3] proposed a nested particle swarm optimization (PSO) framework and applied it to solve a specific multi-leader one-follower nonlinear bilevel decision model for day-ahead electricity markets. The leader (upper level problem) and the follower (lower level problem) were solved by two separate PSO algorithms in an iterative manner. Such a bilevel PSO approach was also applied to solve bi-level pricing problems in supply chains [33], and satisfactory results were obtained. Recently, Zhao and Wei [34] proposed a nested particle swarm algorithm based on sphere mutation to solve bi-level optimization. The simulation results show that the proposed algorithm is effective. Sinha et al [35] presented a nested evolutionary strategy to find an optimal solution for a multi-period multi-leader-follower

Stackelberg competition model, with nonlinear cost and demand functions, and discrete production variables. The strategy was evaluated on a test-suite of bilevel problems, and it has been shown that the method is able to handle difficult bilevel problems. Angelo et al [36] proposed a nested bilevel differential evolution (BlDE), where two DE algorithms are used to solve the upper and the lower level problems. As mentioned earlier, even simple nested methods can be computationally expensive due to the need for evaluating a large number of lower level functions. Recently, a bilevel covariance matrix adaptation evolution strategy (BL-CMA-ES) is proposed in [37], where a novel search distribution sharing mechanism was designed to extract a priori knowledge of the lower-level problem from the upper-level optimizer; it was shown that the time for function evaluations could significantly be reduced through the proposed method. Huang and Wang [38] proposed a new framework (called GO) to identify and utilize the interactions between upper-level and lower-level variables for scalable bilevel optimization problems. However, the identification of the interactions between upper-level and lower-level variables for problems defined in a high dimensional space can be a very challenging task to do.

2.3 Co-evolutionary methods

Co-evolutionary algorithms generally maintain two populations, one for upper level, and another for lower level. The two populations evolve separately, but periodically exchange information to achieve a balanced evaluation for the problem. Oduguwa and Roy [39] firstly proposed a co-evolutionary approach, called bi-level genetic algorithm (BiGA), which can be used for solving bilevel optimization problems. The co-evolutionary operator is used to preserve the interactive nature of the bilevel optimization problem in the search process. An external elite population is maintained to identify the elite members of both populations after the co-evolutionary operation for every generation. However, the algorithm has limited ability to handle constraints, and it can get stuck in a local optimum. Legillon et al [40] further extended the BiGA and developed a more general co-evolutionary bilevel method using repeated algorithms (CoBRA). The application to a bi-level transportation problem with linear objectives and constraints confirmed its effectiveness. Chaabani et al [41] presented a co-evolutionary decomposition based bilevel algorithm (CODBA) to tackle combinatorial BLOPs. To reduce the complexity of the low level task, the lower level population is decomposed into M well distributed subpopulations over the search space for the lower level problem, and the sub-populations co-evolve in parallel using M threads (one thread for each sub-population). Recently, inspired from chemical reaction optimization algorithm, Chaabani et al. [42] proposed a new co-evolutionary decomposition algorithm, called E-CODBA (Energy-based CODBA), to solve combinatorial bi-level problems. However, the co-evolution in CODBA and E-CODBA is only limited to the lower population, and is different from the BiGA [39] and CoBRA [40], where the co-evolution is applied to both the upper and lower populations. So strictly speaking, the CODBA and E-CODBA are actually a nested approach. A similar co-evolutionary framework was reported in [43], where the upper and lower levels are in a nested structure but the lower level follows a co-evolutionary scheme of two EAs to reduce the computational cost of obtaining feasible solutions. Recently, by introducing a migration scheme and defining two populations in each level, Said et al [44] presented a Co-Evolutionary Migration-Based Algorithm (CEMBA) to solve combinatorial bi-level optimization problems. CEMBA has been validated on a set of bi-level combinatorial production-distribution planning benchmark instances. However, similar to CODBA, the individual migration in CEMBA is only limited to the upper or lower level population, thus CEMBA is essentially a nested approach.

2.4 Surrogate model assisted methods

In the last few years, Sinha and co-workers have developed a series of efficient bilevel evolutionary algorithms [11], [12], [14], [18]-[21]. For example, to reduce the lower level function evaluations and improve the computational efficiency, a novel and efficient bilevel evolutionary algorithm based on quadratic approximations (BLEAQ) of optimal lower level variables with respect to the upper level variables was proposed in [11]. To further enhance the performance of BLEAQ, an improved version was introduced by incorporating archiving and local search [14]. The archive is used to store the feasible members produced so far, therefore a larger pool of members for better quadratic approximations of optimal lower level solutions is available. Moreover, Sinha et al. [45] proposed a modified version of BLEAQ to reduce the computational expense by iteratively approximating the lower level rational reaction mapping and the lower level optimal value function mapping. However, the quadratic approximation is time demanding especially when the problem is defined in a high dimensional space and the archive is large, making the approximation accuracy of quadratic programming become lower with the increasing of the upper level dimensions. Recently, Islam et al. [46] presented a surrogate assisted approach for single-objective bilevel optimization (SABLA), which uses multiple surrogates such as response surface models of orders 1 and 2, and Kriging to approximate the lower level objective/constraint functions. To further reduce the computational expense of the upper level problem, Islam et al. [47] proposed an improved SABLA (SA-SA) which uses surrogate-assisted search at both levels to solve bilevel problems. However, the training of multiple surrogate models is computationally expensive and the algorithm realization is complicated.

Angelo et al. [17] proposed a bilevel differential evolution framework assisted by a simple *k*-NN based surrogate model. The method uses two nested DE algorithms: one for the low level optimization task and another for upper level task. The *k*-NN based surrogate model is chosen to replace the lower level optimization on a specific probability. However, the results indicated that the associated surrogate models may be too simple to efficiently solve the variety of test problems as demonstrated in [17], where it showed that when the surrogate model was used at a probability larger than 0.5, the method generated poor quality solutions in some cases, and the convergence of the upper level was compromised.

3. The proposed approach

This work mainly focuses on the evolutionary algorithms for single-objective bilevel optimization. Since the bilevel optimization is nested in nature, the development of nested approach is a simple and intuitive choice for most applications. However, nested approaches are computationally expensive because a corresponding lower level optimization procedure must be performed for each upper level decision vector. So a nested approach requires an efficient method for solving the lower level task producing good responses for the upper level task, and therefore improving the overall efficiency of the bilevel optimization process. When it comes to the framework with EAs in both levels, a way to improve the computation efficiency is to reduce the load of function evaluations in the lower level. Obviously, a small population size, local oriented operators and less evolution generations are beneficial to cut down the function evaluations. In this paper, a bilevel differential evolution with *k*-NN approximation and self-adaptive strategies is proposed to achieve these objectives. The proposed approach follows a nested structure, where the *k*-NN approximation and self-adaptive strategies are used to reduce the lower level function evaluations and this is achieved by an efficient lower level DE.

3.1 *k*-NN based approximation

One of the difficulties in bilevel optimization is the nested lower level optimization problem where the upper level constraint is required to be solved for each of the upper decision vectors. If the lower level optimization problem can be approximated by a simple model, the efficiency of the bilevel optimization can be greatly improved. To reduce the function evaluations on both levels, a *k*-NN approximation assisted DE was proposed in [17]. In this work, while the *k*-NN based surrogate model is also used to approximate the lower level problem, the approximation strategy proposed here is completely different from that given in [17] which is actually a probability based mechanism.

For each individual of the upper initial population, the lower level DE (LLDE) is invoked to solve the lower level problem. If the obtained lower level solution satisfies the lower level constraints, the feasible lower level vector and the corresponding upper decision vector are stored into an external archive. But if the obtained lower level solution is infeasible, it will not be included in the archive. In the later evolutions, the lower level feasible solutions obtained by the LLDE and the corresponding upper level vectors are also added to the archive.

Consider a new upper level candidate $\boldsymbol{x}^u$ and the archive $A = \{(\boldsymbol{x}_1^u, \boldsymbol{x}_1^l), \cdots, (\boldsymbol{x}_N^u, \boldsymbol{x}_N^l)\}$, the *j*th lower level variable is approximated by the following inverse distance weighting (IDW) interpolation function:

$$\tilde{x}_j^l(\boldsymbol{x}^u) = \begin{cases} \frac{\sum_{i=1}^k w_i(x^u) x_{i,j}^l}{\sum_{i=1}^k w_i(x^u)}, & \text{if } d(\boldsymbol{x}^u, \boldsymbol{x}_i^u) \neq 0 \\ x_i^l, & \text{if } d(\boldsymbol{x}^u, \boldsymbol{x}_i^u) = 0 \end{cases}, i = 1, \dots, k \quad (5)$$

where

$$w_i(\boldsymbol{x}^u) = \frac{1}{d(x^u, x_i^u)^p} \quad (6)$$

is a simple IDW weighting function, which was initially defined by Shepard [48], $\boldsymbol{x}^u$ denotes an interpolated (arbitrary) point, $\boldsymbol{x}_i^u$ is an interpolating (known) point, *d* is a distance metric operator measuring the distance from the known point $\boldsymbol{x}_i^u$ to interpolated point $\boldsymbol{x}^u$, *p* is a positive real number, called the power parameter, and *k* is the number of nearest neighbors used in interpolation. Obviously, the weight *w* decreases with the increase of the distance *d*. Greater values of *p* assign greater influence to when the two points $\boldsymbol{x}^u$ and $\boldsymbol{x}_i^u$ becomes closer. This study uses the Euclidean distance metric, that is, $p = 2$.

For a newly generated upper vector by evolution operators, a number of *k* nearest neighbors in the archive are identified to approximate the lower level optimal variables using the IDW function. If the new upper vector is very close to its nearest archived member, the estimated lower level solution is then used to substitute the corresponding optimal solution. Therefore, it is not necessary to carry out a lower level DE. If the new upper vector is not sufficiently close to any one of the members in the archive, the approximated solution can then only be treated as a good prediction of the lower level optimum. In this case, the approximation is used as an interim solution to generate an initial population for the LLDE. Obviously, the initialization strategy will let the LLDE search around the potential lower level optimum, and therefore the convergence is speeded up. Hence, whatever the *k*-NN approximation accuracy, it is useful to help improve the efficiency of LLDE.

3.2 Self-adaptive strategies for lower level population

A general self-adaptive control ratio based on the nearest archived neighbor distance is proposed to adjust the lower level population size and search radius. For a newly generated candidate for the upper level problem, if it is necessary to run LLDE to obtain the lower level optimum, the lower population size and search radius are scaled to its nearest archived neighbor distance. A smaller nearest neighbor distance means a higher similarity of two upper vectors, as well as the closer of the two corresponding lower level optima. Hence, a small population size and a small search radius are assigned to the LLDE to reduce the workload if the lower level function evaluations if the newly generated upper candidate has a small distance to its archived nearest neighbor, and vice versa. Taking a 2D case as an example (shown in Fig. 1), the nearest neighbor distance based self-adaptive control ratio δ is defined as:

$$\delta = \left(\frac{d_{nn}}{d_{bs}}\right)^{\frac{1}{c}} \tag{7}$$

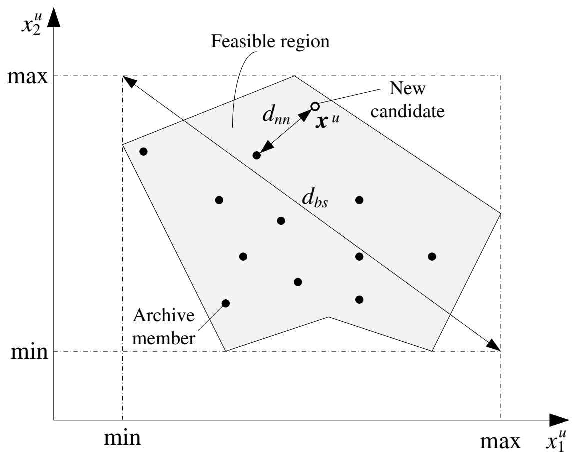


Fig.1 Calculation of self-adaptive control ratio δ in 2D space

where d_{nn} is the Euclidean distance between the new upper candidate and its archived nearest neighbor, d_{bs} is the maximum distance of the upper level search space, and $c \geq 1$ is an integer constant. Note that d_{nn} is always smaller than or equal to d_{bs}, so $\delta \leq 1$, and a larger d_{nn} means a larger δ. Hence, δ is a normalized index to measure the similarity between the upper candidate and the archived nearest member. A smaller δ means the upper candidate is more similar to the archived closest member, and vice versa.

A. self-adaption of the lower level population size

Based on an appropriately specified self-adaptive control ratio δ, the lower level population size NP^l is dynamically adjusted using the following strategy:

$$NP^l = \max([\delta NP^l(0)], NP_{min}^l) \tag{8}$$

where $NP^l(0)$ is the initial lower level population size, NP_{min}^l is the minimum lower level population size to be used, and $[\cdot]$ denotes the greatest integer function.

It is well known that the population size has great influence on the DE's performance. To avoid the deterioration of the effectiveness of LLDE caused by too small population size, the lower level population size is confined between a minimum population size and the initial population size. Here, the minimum population size NP_{min}^l is determined by the dimensions of lower level optimization problems, and the rule is :

$$NP_{min}^l = \begin{cases} 3D^l, & \text{if } D^l \leq 5 \\ 0.5NP^l(0), & \text{else} \end{cases} \tag{9}$$

where D^l is the number of variables involved in the lower level problem. For problems with lower dimension ($D^l \leq 5$), NP_{min}^l can be chosen to be three times of the dimension, while it is a half of the initial population size for higher dimensional problems.

To keep a comparative large population size for LLDE when there is a small distance between the upper candidate and the closest archived member, a self-adaptive control ratio δ with a large parameter c is desirable. In this paper, c is set to be 10 for the self-adaptation of the lower population size.

B. self-adaption of the lower level search radius

Similar to the population size, a self-adaptive strategy for the lower level search radius is as below:

$$\gamma_j^l = \max(\delta, 0.01) \cdot (x_{max,j}^l - x_{min,j}^l) \tag{10}$$

where $x^l_{max,j}$ and $x^l_{min,j}$ are the upper and lower bounds of the *j*th lower level variable. To enhance local search, a small power parameter of *c* = 3 is used. In addition, a minimum radius ratio of 0.01 is introduced to avoid too small search radius. Obviously, a higher similarity of an upper candidate to an archived member would mean a smaller search radius for the LLDE. Therefore, the convergence rate of the LLDE can be improved accordingly.

For a newly generated upper candidate $\boldsymbol{x}^u$, if the LLDE cannot be sufficiently approximated by the *k*-NN approximation $\tilde{\boldsymbol{x}}^l$, the corresponding population size of LLDE should be determined by Eq. 9 first and the initial population can then be generated based on the approximated $\tilde{\boldsymbol{x}}^l$ and the search radius defined by (10). As a result, the LLDE will do local search around the $\tilde{\boldsymbol{x}}^l$, and the efficiency can thus be enhanced. The lower initial population can be generated by

$$x^l_{ij}(0) = \tilde{x}^l_j + \gamma^l_j \cdot Gauss(0,1), i = 1, \cdots, NP^l, j = 1, \cdots, D^l \tag{11}$$

where *Gauss*(0,1) represents the Gaussian distribution with the mean value 0 and the standard deviation 1.

3.3 Algorithm framework

For bilevel optimization, both the computational efficiency and solution robustness should be equally emphasized. Differential Evolution proposed by Storn and Price in 1995 [16] is a simple but powerful real-coded stochastic optimization algorithm. In the last decades, DE has been successfully applied in many practical cases due to its good properties, and has been proven to be one of the most powerful global numerical optimization algorithms in the evolutionary algorithm community. In recent years, various mutation operators have been proposed [49]. Generally, different mutation operators have different features and there is no single one that can always perform well for all types of problems [50]. Among them, the DE/rand/1 and DE/best/1 are two of the most frequently used mutation strategies. To improve DE's performance, the ensemble of multiple mutation operators is one of the popular approaches [51]. In this work, three mutation operators are selected to develop an efficient and effective bilevel DE algorithm.

For the upper DE procedure, a simple ensemble mutation operator by probabilistically selecting DE/rand/1 or DE/best/1 is constructed to balance the global exploration and local search. The DE/best/1 and DE/rand/1 are selected based on a specified probability. To enhance the local search efficiency, a higher probability ($\tau = 0.7$) is assigned to the DE/best/1, and the DE/rand/1 is used as an assistant strategy to improve the global convergence ability. The pseudo-code of the simple ensemble mutation operator is illustrated in Algorithm 1. Where *rand* is a uniformly distributed random number within [0, 1], F_r and F_b are mutation constants for DE/rand/1 strategy and DE/best/1 strategy, respectively.

Algorithm 1 The Proposed Ensemble Mutation Operator

```
    Input: $NP^u$, $F^u_b$, $F^u_r$, $CR^u_b$, $CR^u_r$, $D^u$, and index $i$
1:  Select randomly $r1 \neq r2 \neq r3 \neq i$
2:  $j_{rand}$ = rndint(1, $D^u$)
3:  if $rand \leq \tau$ then /* DE/best/1 mutation strategy */
4:      $\boldsymbol{v}^u_i = \boldsymbol{x}^u_{best} + F^u_b \cdot (\boldsymbol{x}^u_{r1} - \boldsymbol{x}^u_{r2})$
5:      for $j$ = 1 to $D^u$ do
6:          if $rand < CR^u_b$ or $j = j_{rand}$ then
7:              $u^u_{ij} = v^u_{ij}$
8:          else
9:              $u^u_{ij} = x^u_{ij}$
10:         end if
11:     end for
12: else
13:     $\boldsymbol{v}^u_i = \boldsymbol{x}^u_{r1} + F^u_r \cdot (\boldsymbol{x}^u_{r2} - \boldsymbol{x}^u_{r3})$
14:     for $j$ = 1 to $D^u$ do
15:         if $rand < CR^u_r$ or $j = j_{rand}$ then
16:             $u^u_{ij} = v^u_{ij}$
17:         else
```

```
18:         $u_{ij}^u = x_{ij}^u$
19:       end if
20:     end for
21: end if
    Output: $\boldsymbol{u}_i^u$
```

For the lower level DE procedure, the DE/target-to-best/1 and DE/best/1 are dynamically selected based on the similarity between the upper candidate and the archived members. If a newly generated upper level candidate is close to an upper level member in the archive, it is often expected that their lower level solutions are also close to each other, therefore, the DE/best/1 is used to do local search around the archived lower level solution to speed up the convergence. Otherwise, the DE/target-to-best/1 is selected to solve the lower level optimization problem since it has a good balance between the exploration and exploitation. But for the upper initial population, the DE/target-to-best/1 is always used to obtain the lower level optima. The above self-adaptive mutation strategy for lower level DE is illustrated in Algorithm 2, where $\bar{d}(0)$ is the average distance among initial individuals in upper population.

Algorithm 2 The Self-adaptive Mutation Strategy

```
    Input: $d_i^{nn}$, $\bar{d}(0)$ and iteration t
1: If $d_i^{nn} \geq 0.5\bar{d}(0)$ or $t == 0$ then
2:   flag = 0 /*DE/target-to-best/1 mutation strategy*/
3: else
4:   flag = 1 /*DE/best/1 mutation strategy*/
5: end if
    Output: flag
```

In the following, the detailed procedures of the bilevel DE with *k*-NN approximation and self-adaptive control strategies are described.

A. Upper level optimization

The upper level optimization starts with a randomly initialized population in the upper variable space. For each of the initial individuals, the lower level DE is used to solve the lower optimization problem, and the obtained lower level optimal solutions and the corresponding upper level solutions are stored into an external archive *A*. By evaluating the upper level objective function and constraints, the upper level fitness is then assigned. Thereafter, the upper evolution operators are applied to the upper population and new candidates are generated. For each of the candidates, the *k*-NN approximations are firstly applied to estimate the lower optimal variables based on the archived members. If the nearest distance between the candidate and some of the archived members is small enough ($d_i^{nn} \leq d_{bs} \cdot 10^{-5}$), the approximated lower level variables are directly accepted as the lower level optimal solution. Otherwise, the approximated lower level solution is used as the base solution to generate an initial population for the lower level DE. Moreover, the self-adaptive strategies based on the nearest distance for the lower population size and search radius are used to enhance the convergence performance of the lower level DE. The pseudo-code of upper level optimization is shown in Algorithm 3.

Algorithm 3 Upper Level Differential Evolution (ULDE)

```
     Input: $NP^u$, $F_b^u$, $F_r^u$, $CR_b^u$, $CR_r^u$, $T^u$, and $\alpha_{stop}^u$
 1: Generate the initial population randomly in the upper search space
 2: Set flag=0 to call LLDE with DE/target-to-best/1 mutation
 3: Store the initial solutions $(\boldsymbol{x}^u(0), \boldsymbol{x}^{l*}(0))$ into the archive A
 4: Evaluate the upper fitness $fit^u(\boldsymbol{x}^u(0), \boldsymbol{x}^{l*}(0))$ for each initial
    solution based on upper function and constraints
 5: Set iteration counter t = 1
 6: while the stop criteria are not satisfied do
 7:   for i =1 to $NP^u$ do
 8:     Execute Algorithm 2 to obtain trail vector $\boldsymbol{u}_i^u$
 9:   end for
10:   for i =1 to $NP^u$ do
11:     Calculate distances between $\boldsymbol{u}_i^u$ and the archive members
```

Accepted by *Memetic Computing* on 5 March 2021. https://doi.org/10.1007/s12293-021-00335-8

```
            and identify the nearest distance $d_i^{nn}$ and $k$ nearest archive
            neighbors
12:         Estimate the optimal lower level variables using $k$-NN
            approximation for $\boldsymbol{u}_i^u$
13:         if $d_i^{nn} \leq d_{bs} \cdot 10^{-5}$ then
14:           Accept the estimated solution as the optimal lower solution
15:         else
16:           Determine the population size $NP_i^l$ of LLDE according to
              Eq.8 and Eq.9
17:           Calculate the search radius of LLDE according to Eq.10
18:           Initialize the $Pop_i^l$ according to Eq.11
19:           if $d_i^{nn} \leq 0.5\bar{d}(0)$
20:             *flag*=1, execute LLDE with DE/best/1mutation
21:           else
22:             *flag*=0, execute LLDE with DE/target-to-best/1mutation
23:           end if
24:           copy the optimal lower level solution $\boldsymbol{x}_i^{l*}$ to the archive
25:         end if
26:         Evaluate the fitness of candidate $(\boldsymbol{u}_i^u, \boldsymbol{x}_i^l)$ based on upper
            level function and constraints
27:         if $\mathrm{fit}_i^u(\mathrm{u}_i^u, \mathrm{x}_i^{l*})$ is better than or equal to $\mathrm{fit}_i^u(\mathrm{x}_i^u, \mathrm{x}_i^l)$ then
28:           $(\boldsymbol{x}_i^u, \boldsymbol{x}_i^l) = (\mathrm{u}_i^u, \mathrm{x}_i^{l*})$
29:         end if
30:     end for
31:     $t = t+1$
32: end while
    Output: $(\boldsymbol{x}^{u*}, \boldsymbol{x}^{l*})$
```

B. Lower level optimization

If the estimated lower level variables are not accurate enough to approximate the optimal lower level solution, the lower level DE will be executed. For the nested BLEA structure, an efficient lower level EA is required. Hence, the two mutation strategies, DE/target-to-best/1 and DE/best/1, are used to develop an efficient lower level DE, and in each generation only the one that generates the 'best' individual is used. Moreover, the lower level population size and search radius are dynamically adjusted with respect to the similarity between the upper candidates and the archived members; this is beneficial to reduce the lower level function evaluations. The pseudo-code of the lower level DE is shown in Algorithm 4.

Algorithm 4 Lower Level Differential Evolution (LLDE)

```
    Input: $Pop^l$, $\boldsymbol{x}^u$, *flag*, $F^l$, $CR^l$, $T^l$, and $\alpha_{stop}^l$
1:  Evaluate the lower fitness $fit^l(\boldsymbol{x}^u, \boldsymbol{x}^l(0))$ for each lower initial
    individual based on lower function and constraints
2:  Set iteration counter $t = 1$
3:  while the stop criteria are not satisfied do
4:    for $i$ =1 to $NP^l$ do
5:      Select randomly $r1 \neq r2 \neq i$
6:      $j_{rand}$ = rndint(1, $D^l$)
7:      if *flag* == 0 /* DE/target-to-best/1*/
8:        $\boldsymbol{v}_i^l = \boldsymbol{x}_i^l + F^l \cdot (\boldsymbol{x}_{best}^l - \boldsymbol{x}_i^l) + F^l \cdot (\boldsymbol{x}_{r1}^u - \boldsymbol{x}_{r2}^u)$
9:      else /* DE/best/1*/
10:       $\boldsymbol{v}_i^l = \boldsymbol{x}_{best}^l + F^l \cdot (\boldsymbol{x}_{r1}^u - \boldsymbol{x}_{r2}^u)$
11:     end if
12:     for $j$ = 1 to $D^l$ do
13:       if *rand* < $CR^l$ or $j = j_{rand}$ then
14:         $u_{ij}^l = v_{ij}^l$
15:       else
16:         $u_{ij}^l = x_{ij}^l$
17:       end if
18:     end for
19:   end for
20:   for $i$ =1 to $NP^l$ do
21:     Evaluate the fitness of candidate $(\boldsymbol{x}^u, \boldsymbol{x}_i^l)$ based on lower
22:     level function and constraints
```

```
        if $fit_i^l(\boldsymbol{x}^u, \boldsymbol{u}_i^l)$ is better than or equal to $fit_i^l(\boldsymbol{x}^u, \boldsymbol{x}_i^l)$ then
23:         $\boldsymbol{x}_i^l = \boldsymbol{u}_i^l$
24:     end if
25:   end for
26:   $t = t+1$
27: end while
28: Output: $\boldsymbol{x}^{l*}$
```

C. Flowchart of SABiLDE

The overall flowchart of the bilevel DE with k-NN approximation and self-adaptive control strategies is shown in Fig.2. In the ULDE, a nested LLDE is always invoked for each upper level initial individual. But for each upper trial individual, if it is similar to one of the history individuals in the archive, the k-NN is used to approximate the lower level optimal solution. Moreover, the population size and search radius of the LLDE are adaptively adjusted to reduce the function evaluation based on the nearest neighbor distance d_{nn}.

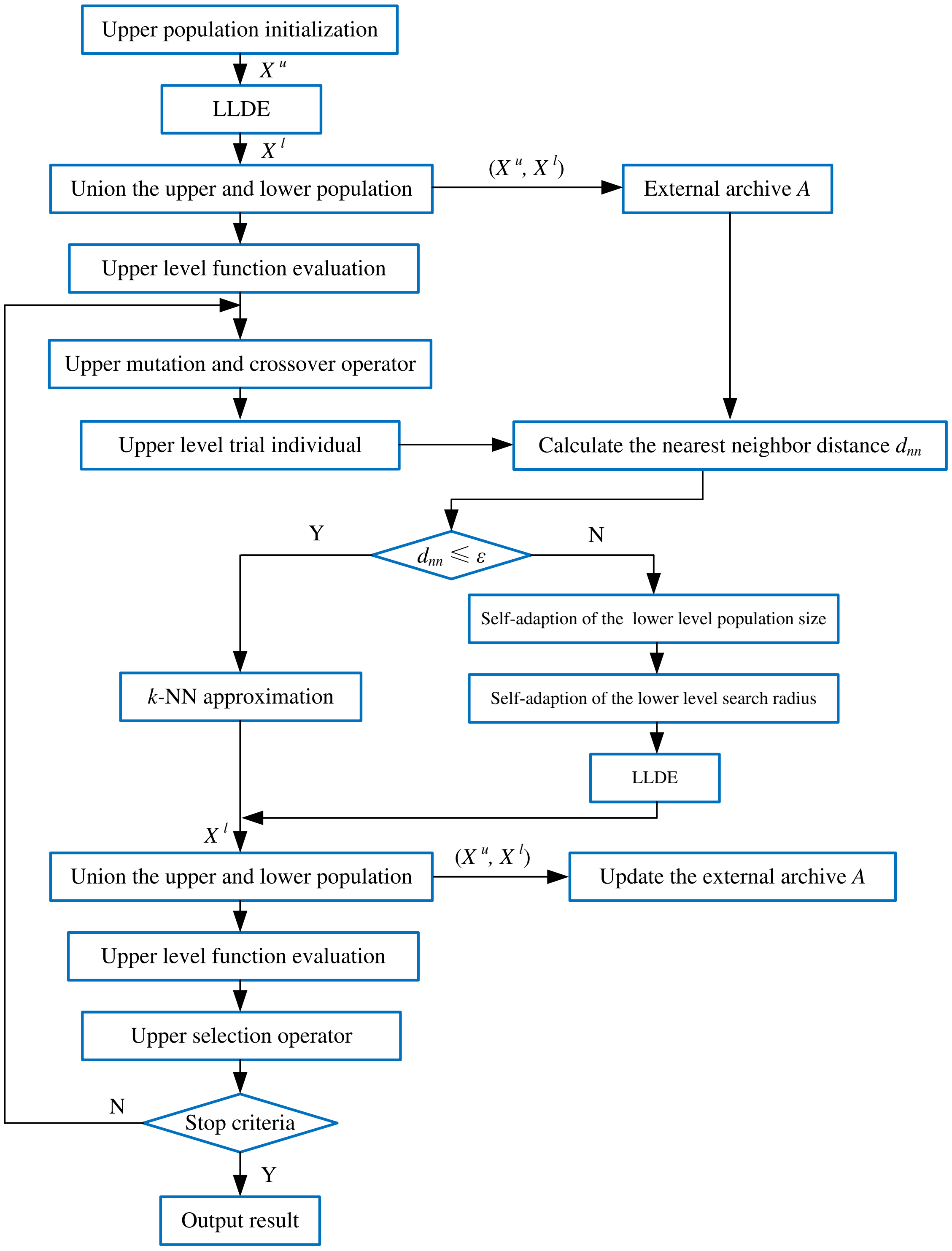

Upper population initialization
X^u
LLDE
X^l
Union the upper and lower population
(X^u, X^l)
External archive A
Upper level function evaluation
Upper mutation and crossover operator
Upper level trial individual
Calculate the nearest neighbor distance d_{nn}
Y
$d_{nn} \leqslant \varepsilon$
N
Self-adaption of the lower level population size
k-NN approximation
Self-adaption of the lower level search radius
LLDE
X^l
Union the upper and lower population
(X^u, X^l)
Update the external archive A
Upper level function evaluation
Upper selection operator
N
Stop criteria
Y
Output result

Fig. 2 Flowchart of the proposed SABiLDE

4. Numerical results and analysis

Two sets of test problems are chosen to evaluate the performance of the proposed SABiLDE. The first set includes 10 standard bilevel test problems (referred to as TP [11]) collected from different sources. The second set is the recently proposed SMD test suite [18], which consists of 12 scalable bilevel problems with different difficulties in terms of convergence at the two levels, complexity of interactions between the two levels and multi-modalities at each of the levels. Among the 12 problems, the first eight are unconstrained and the remaining four are constrained.

The results of the proposed algorithm are compared with the following four representative BLEAs for generic bilevel optimization problems: SABLA [46], BL-CMA-ES [37], BLEAQ [14] and BlDE [16],. The BlDE is an improved version of the nested bilevel differential evolution discussed in [35], with a k-NN based surrogate model being incorporated in the lower optimization. The BLEAQ proposed in [14] is an efficient bilevel evolutionary algorithm based on quadratic approximations of the lower level optimal variables as a function of upper level variables, which has been shown to outperform a number of contemporary strategies for bilevel optimization. The BL-CMA-ES [37] is a newly developed evolutionary bilevel optimization algorithm based on covariance matrix adaptation. The SABLA is a surrogate assisted approach for single-objective bilevel optimization, which uses multiple surrogates to approximate the lower level solutions.

4.1 Parameter setting and test platform

The population sizes at both levels are set to be 30 for both the TP group and the SMD suite. The mutation scaling factor F and crossover rate CR for DE/rand/1 mutation strategy are 0.5 and 0.1, respectively. The two strategies, DE/best/1 and DE/target-to-best/1, have the same mutation scaling factor and crossover rate, i.e. $F = 0.5$ and $CR = 0.9$. The values for the upper stop criterion α^{u}_{stop} and the lower stop criterion α^{l}_{stop} are both 1E-6. For another stop criterion, the maximum generations of no improvement are set to be 20 for both levels. The number of nearest neighbors of k used in interpolation is determined by

$$k = \min\{2^{D^u} + 1, (D^u + 1)(D^u + 2)/2, NP^u\} \tag{18}$$

where D^u is the dimension of the upper level decision vector, NP^u is the upper level population size. To highlight the local interpolation and reduce computational cost of the approximation procedure, k cannot be larger than the upper level population size.

For BLDE, the probability of using the k-NN based surrogate model is 30%, and k is set to 2 as suggested in [16]. Other parameter settings of the compared algorithms are determined according to their original references. The code for the algorithm is realized in MATLAB 2012a, and all the computations have been performed on a laptop with 64 bit Windows 10 platform, 2.6GHz double-core Intel Core i5 processor and 4GB of 1600MHz DDR3 RAM. The MATLAB code of BLEAQ is available at http://bilevel.org. We rewrote the code of BlDE based on the pseudo-codes in [16]. To avoid randomness, each test function is optimized over 30 independent runs. For fair comparison, a total of 30 different initial populations were considered, starting from which each algorithm was run 30 times, and the relevant overall performance was then compared. Because the code of SABLA and BL-CMA-ES are not open access, we just copy the results in the literature therein.

To test the performance of the proposed algorithm, the test problems SMD1 to SMD9 with 10 dimensions are considered. For problems SMD 1-5, p=3, q=3, and r=2; for problems SMD 7-9, p=3, q=1, r=2; for problem SMD6, p=3, q=1, r=2, and s=2, where p, q, r, and s are the parameters of SMD test suite [18]. For the other three test problems SMD10 to SMD12 with complex constraints, only the case of 5 dimensions is considered since the three algorithms (BlDE, SABLA and BL-CMA-ES) all fail to solve the three test problems with 10 dimensions. For the three problems with 5 dimensions, the values of the three parameters p, q and r are set to be 1, 2, and 1, respectively.

4.2 Performance evaluation criteria

Performance of the algorithms used is evaluated and compared using the following three criteria.

1) Median of error values: the median of the errors between the obtained solutions and the true optimum over all independent runs is used to measure the accuracy of the solutions obtained.

2) Success rate (SR): the percentage of convergence to the optimal solution in all independent runs are used to evaluate the robustness and reliability of the proposed algorithm.

3) CPU running time: the average and minimum CPU running time to reach the optimal solution is used to assess the computational cost.

4.3 Comparison with the state-of-the-art

A. Solutions

The median solution accuracy of SABiLDE, SABLA, BL-CMA-ES, BLEAQ and BlDE are shown in Table 1 for the TP test problems and the SMD test suite (SMD1 to SMD9 with 10 dimensions, SMD10 to SMD12 with 5 dimensions). If the difference between the function value achieved by an algorithm and the true optimal function value is no more than 0.1, it is considered that the test problem is solved [18]. The success rates of the three algorithms in 30 runs for the two test sets are also shown in Table 1. In order to highlight the overall best result, the significantly better values are marked in bold. While reporting the median errors, we consider a precision of 1E-6. Consequently, if the error is less than 1E-6, it is simply set to 1E-6. This is can facilitate the comparison of extremely small values, where the difference in values is not significant.

Table 1 Solution comparison among SABiLDE, SABLA, BL-CMA-ES, BLEAQ and BlDE

Prob	SABiLDE			SABLA			BL-CMA-ES			BLEAQ			BlDE		
	UL Acc.	LL Acc.	SR(%)	UL Acc.	LL Acc.	SR(%)	UL Acc.	LL Acc.	SR(%)	UL Acc.	LL Acc.	SR(%)	UL Acc.	LL Acc.	SR(%)
tp1	8.85E-4	1.17E-3	100	**1.00E-6**	**1.00E-6**	100	1.11E-6	3.78E-5	100	1.38E-6	**1.00E-6**	100	5.33E-2	1.13E-1	83.3
tp2	6.42E-5	1.83E+0	86.7	1.34E-6	**3.53E-6**	100	**1.00E-6**	2.23E-4	100	1.10E-3	9.99E+1	36.7	1.00E-6	8.81E+1	36.7
tp3	2.89E-5	9.34E-5	100	**1.00E-6**	**1.00E-6**	100	2.53E-2	2.80E-2	100	1.09E-5	2.50E-5	100	8.96E-5	2.41E-5	100
tp4	3.93E-4	1.29E-6	100	8.21E+2	1.00E-6	**-**	**1.00E-6**	**1.00E-6**	100	1.64E-3	5.17E-4	100	6.54E-4	1.02E-5	100
tp5	**3.65E-6**	**1.92E-6**	100	4.46E-6	7.77E-5	100	1.73E-1	2.20E-1	-	3.03E-3	1.51E-2	80	1.16E-5	8.37E-5	76.7
tp6	7.77E-4	2.78E-3	100	1.75E-4	**1.00E-6**	100	**1.00E-6**	2.48E-5	100-	7.77E-4	2.78E-3	100	7.80E-4	2.80E-3	100
tp7	8.01E-4	6.92E-4	100	**1.00E-6**	**1.00E-6**	100	6.74E-4	6.74E-4	100	7.84E-4	7.84E-4	100	8.46E-4	1.66E-3	100
tp8	**1.00E-6**	**1.00E-6**	100	2.42E-5	**1.00E-6**	100	**1.00E-6**	1.05E-4	100	6.11E-3	9.99E+1	46.7	1.00E-6	9.48E+1	36.7
tp9	**1.00E-6**	**1.00E-6**	100	6.51E-6	1.72E-5	100	**1.00E-6**	**1.00E-6**	100	8.94E-5	**1.00E-6**	100	1.80E-3	**1.00E-6**	100
tp10	2.58E-6	**1.00E-6**	100	2.63E-6	4.65E-4	100	**1.00E-6**	**1.00E-6**	100	2.39E-5	**1.00E-6**	100	1.62E-3	**1.00E-6**	100
SMD1	**1.00E-6**	**1.00E-6**	100	**1.00E-6**	**1.00E-6**	100	**1.00E-6**	**1.00E-6**	100	2.54E-5	1.74E-5	100	2.39E-5	1.36E-5	100
SMD2	**1.00E-6**	**1.00E-6**	100	**1.00E-6**	**1.00E-6**	100	**1.00E-6**	1.28E-6	100	1.39E-4	2.22E-4	100	4.18E-4	3.75E-3	66.7
SMD3	**1.00E-6**	**1.00E-6**	100	**1.00E-6**	**1.00E-6**	100	**1.00E-6**	1.12E-6	100	2.78E-2	9.29E-4	100	1.90E-5	1.68E-5	100
SMD4	**1.00E-6**	**1.00E-6**	100	**1.00E-6**	**1.00E-6**	100	**1.00E-6**	8.18E-6	100	2.45E-4	2.06E-4	100	8.99E-1	9.55E-1	3.33
SMD5	**1.00E-6**	**1.00E-6**	100	**1.00E-6**	**1.00E-6**	100	**1.00E-6**	1.49E-6	100	9.34E-5	3.22E-4	100	9.54E-5	1.61E-4	83.3
SMD6	3.78E-4	2.71E-5	100	**1.00E-6**	**1.00E-6**	100	**1.00E-6**	**1.00E-6**	100	**1.00E-6**	**1.00E-6**	100	1.07E+0	5.02E-1	0.0
SMD7	1.87E-4	1.52E+1	93.3	**1.00E-6**	**1.00E-6**	100	4.91E-2	6.25E+1	-	9.77E-2	1.20E+2	63.3	4.50E+0	5.03E+2	0.0
SMD8	**1.00E-6**	3.58E-5	100	2.76E-3	6.77E-4	100	**1.00E-6**	**1.00E-6**	100	8.81E-3	1.02E-2	90	4.36E-4	3.21E-4	76.7
SMD9	**1.00E-6**	**1.00E-6**	100	1.78E-6	1.77E-6	100	**1.00E-6**	1.98E-6	100	1.64E+0	5.53E+0	16.7	1.58E+0	1.99E+0	0.0
SMD10	**1.00E-6**	8.32E-5	100	3.80E-6	**3.90E-6**	100	1.60E+1	1.00E-6	-	7.94E-1	9.53E-1	0.0	7.11E+0	8.11E+0	0.0
SMD11	3.76E-3	4.54E-3	100	2.31E-1	3.25E-1	**-**	2.66E-3	3.49E-3	100	**3.26E-4**	**7.14E-4**	100	2.92E-1	3.28E-1	0.0
SMD12	6.94E-5	1.10E-4	100	**1.00E-6**	**1.00E-6**	100	1.00E-6	1.07E+1	-	4.51E-1	1.75E+0	0.0	1.98E+0	8.02E+0	0.0

It can be seen from Table 1, SABiLDE obtained better convergence accuracy than BlDE for almost all the test problems. Although both SABiLDE and BlDE use the *k*-NN approximation as the lower level surrogate model to reduce the lower level function evaluations, their assistant mechanism is completely different. BlDE adopts a probability based selection strategy, that is, the *k*-NN approximation is chosen to replace the lower level DE in a specified probability no matter the solution accuracy obtained. However, the *k*-NN based surrogate model is too simple to effectively solve a variety of problems, especially when a small number of neighbors are used (i.e., a small *k* in *k*-NN) for multi-variables interpolation. In BlDE, the number of nearest candidate solutions selected to calculate the lower level variables via the surrogate model is set to 2 ($k=2$). Obviously, the approximation ability of the *k*-NN model used in BlDE is limited. Once a false estimation occurs, the upper optimization could be trapped into a sub-optimum or even a false solution. For example, it can be noticed from Table 1 that BlDE fails to handle most of the test problems, though it is able to converge to the upper and lower true optima of TP3, TP4, TP6, TP7, TP9, TP10, SMD1 and SMD3 with a comparatively high accuracy. In SABiLDE, however, the estimated solution based on *k*-NN approximation is not directly used as the optimal lower level solution but a base solution for the initialization of lower level population if the approximation is not accurate enough. Therefore, the optimality of the lower level solution can be well guaranteed for SABiLDE.

In comparison with the modified BLEAQ, SABiLDE provides better or competitive results for all the test problems except for SMD6. The test problem of SMD6 is very difficult to solve, because there is conflict between the two levels [18]. Moreover, the lower level problem is a multi-modal problem. For any given upper level vector, there are an infinite number of global solutions at the lower level. In the entire global solution set, there is only a single lower level point which corresponds to the best upper level function value [18]. SABiLDE is able to find one of lower level optima for any given upper candidate, but it is difficult to converge to the upper optimum with high accuracy because of the multi-modal property of the lower level problem and the conflict between the two levels. Interestingly, BLEAQ can easily handle this problem with very high precision. A reasonable interpretation may be that the local search at upper level supported by the quadratic approximations is especially beneficial to speed up the convergence of the algorithm for this test problem. However, BLEAQ is unable to solve the problems TP2, TP5, TP8, SMD7 in all runs, and fail to solve the constrained problems SMD10 and SMD12, owing to the introduction of infeasible members at the upper level.

When compared to SABLA, a novel multiple surrogate assisted bilevel algorithm [46], SABiLDE has competitive or better solution accuracy for the majority of the test problems. SABLA performs better than SABiLDE on the problems TP1, TP2, TP3, TP6, TP7, and SMD12, but SABiLDE wins when solving the problems TP4, TP5, TP8, TP9, TP10, SMD8, SMD9, SMD10, SMD11. It's interesting that SABLA cannot solve the upper level problem of TP4, and the upper and lower level problem of SMD11.

For BL-CMA-ES, one of lately developed efficient BLEAs, it can be seen from Table 1 that the method generally obtained better upper level solutions on the most test problems than SABiLDE, but SABiLDE is able to obtain better lower level solutions on the majority of the test problems. Moreover, BL-CMA-ES cannot solve the upper problem of SMD10, the lower problems of SMD7 and SMD12, and the upper and lower problem of TP5. Therefore, it can be concluded that SABiLDE generally demonstrates better robustness than BL-CMA-ES.

As it can be seen from Table 1, SABiLDE failed to find the lower level sub-optima of TP2 and SMD7 in some runs. Although the problem TP2 is a bilevel optimization problem with low dimensions, the lower constraints separate the lower feasible region into several parts, solutions are quite easily to be trapped into sub-optima. In addition, there is a conflict between the two levels. For this problem, as shown in the Table 1, SABLiDE, BLEAQ and BlDE cannot solve the lower problem in some runs, but SABiLDE has better global convergence ability and its success rate is much higher than BlDE and BLEAQ. It should be mentioned that both BlDE and BLEAQ gave false lower level solution to this problem, resulting in corresponding false upper level solutions. Moreover, BlDE and BLEAQ showed similar performance when solving the test problems of SMD7 and SMD10. But the overall performance of BLEAQ is much better than that of BLDE. in that false solutions frequently occurred in BlDE. When the lower level population sizes for TP2 and SMD7 were increased to 40, the premature convergence of SABiLDE in lower levels was avoided. However, with the increasing of population size, the function evaluations in the corresponding lower level or upper level are also increased.

To further quantitatively assess the performance of the proposed method, the Wilcoxon signed ranks test and Friedman test [52] were carried out. The calculation of two tests and the associated multiple comparisons were conducted using the KEEL software tool [53]. Tables 2 and 3 show the upper and lower accuracy results when applying Wilcoxon test to detect significant differences for the control algorithm SABiLDE by means of pairwise comparisons, respectively. The symbol "◦" means that the method in the row performs better than the methods that column, and the "•" means that the method in the column improves the method of the row. Upper diagonal of level significance α = 0.1, and lower diagonal level of significance α = 0.05. As it can be seen from Tables 2 and 3, SABiLDE is significantly better than BlDE in both levels at α = 0.05, and significantly better than BLEAQ in the lower level at α =0.05, and better than BLEAQ in the upper level at α =0.1. However, the difference among SABiLDE, BL-CMA-ES and SABLA is not significant, which means that the three algorithms have competitive optimum performance.

Table 2 Wilcoxon test results of upper level accuracy among different BLEAs

	(1)	(2)	(3)	(4)	(5)
SABiLDE (1)	-	96.0	110.0	195.5●	244.0●
BL-CMA-ES(2)	135.0	-	128.5	165.5●	201.5●
SABLA(3)	121.0	102.5	-	193.0●	212.0●
BLEAQ (4)	36.0o	66.0	38.0o	-	171.0
BlDE (5)	7.0o	51.5o	41.0o	82.0	-

Table 3 Wilcoxon test results of lower level accuracy among different BLEAs

	(1)	(2)	(3)	(4)	(5)
SABiLDE (1)	-	118.5	89.5	195.5●	244.5●
BL-CMA-ES(2)	112.5	-	66.5o	166.5●	209.5●
SABLA(3)	163.5	164.5	-	223.5●	231.0●
BLEAQ (4)	35.5o	64.5	29.5o	-	143.5
BlDE (5)	8.5o	43.5o	22.0o	109.5	-

The average Friedman test rankings of the upper and lower results for all the five algorithms are shown in Fig.3. The lower the bar, the better ranking the algorithm obtains. According to Fig.3, it can be seen that SABiLDE has significantly better convergence performance than BlDE and BLEAQ. In comparison with BL-CMA-ES, SABiLDE has a better rank on the lower level convergence performance, but BL-CMA-ES has better convergence performance on the upper level solutions. In comparison with SABLA, SABiLDE has competitive convergence performance on the upper level solutions but exhibits inferior performance on the lower level solutions.

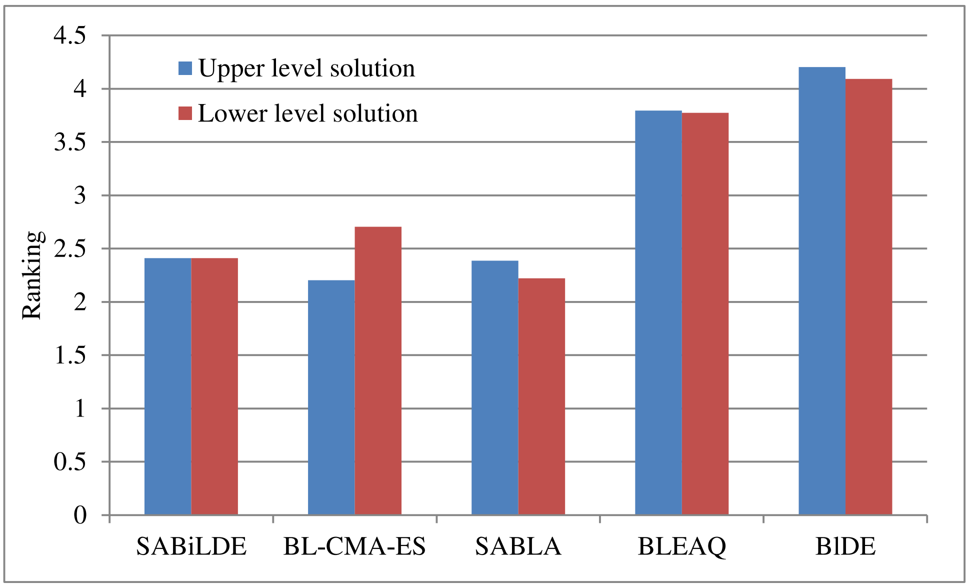


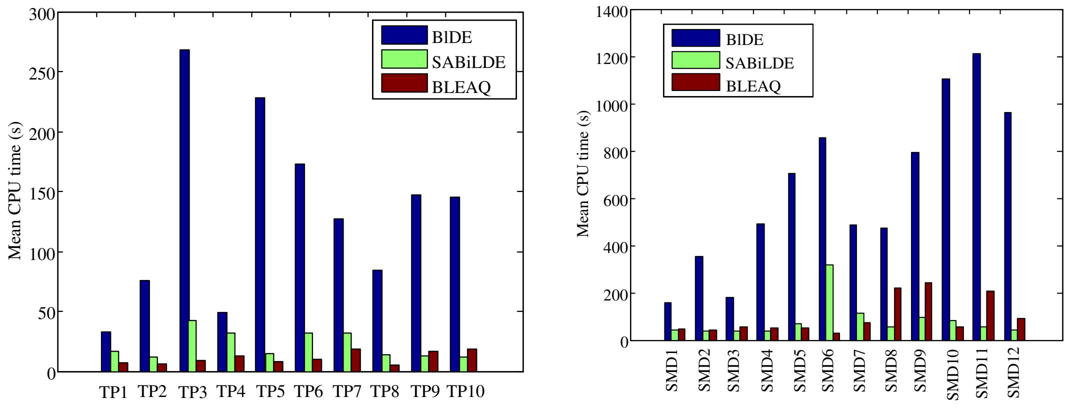
Fig.3 Average Friedman rankings of solution accuracy for different BLEAs

B. Computational efficiency

To compare the computational efficiency, the average and minimum number of function evaluations in both levels are generally used as the performance criterion. However, the number of function evaluations cannot accurately reflect the computational efficiency of BLEAQ [46], because the computational cost of the quadratic approximations of the lower level variables is neglected. If the quadratic approximation procedure is performed, a quadratic function must be constructed for each of the lower variables. That is, for a lower level solution with D^l variables, D^l quadratic approximations should be constructed. Obviously, the computational cost of quadratic approximation should be considered, especially when the dimension of lower level decision vector is high. To roughly measure the computational efficiency of the three algorithms, the CPU running time is recorded in Table 4. Because the code of SABLA and BL-CMA-ES are not open access, it is only compared among SABiLDE, BLEAQ and BlDE. To graphically illustrate the results, the comparison of mean of CPU running time is shown in Table 4 and Fig. 4, from which it can be seen that BLEAQ generally has good efficiency in all of the test problems and outperforms SABiLDE in problems with low dimension such as TP1 to TP8. But when it comes to the problems with high dimension, i.e. TP9, TP10, and SMD1 to SMD9, the superiority of BLEAQ becomes less obvious, and actually SABiLDE provides better or competitive CPU running time in most of the high dimensional problems except for SMD6. The improved efficiency of SABiLDE benefits from the similarity based self-adaptive strategies for the size and initialization of lower population. Fig. 5 demonstrates the overall changing trends of the lower level population size of SABiLDE against the upper evolution when solving the problems of TP1 and SMD1. It can be seen from Fig. 5 that the lower level population size decreases with the increase of the upper population evolution. Obviously, a smaller population size means a smaller number of function evaluations. Moreover, the introduction of the two stop criteria is beneficial to reduce redundant function evaluations in both levels. For the complex constraint problems with low dimension such as SMD10, SMD11, and SMD12, BLEAQ also lost its dominant position on computational efficiency. The possible reason is that the quadratic functions cannot well approximate the complex interactive relationships between the upper and lower variables, and the lower level EA has to be more frequently performed.

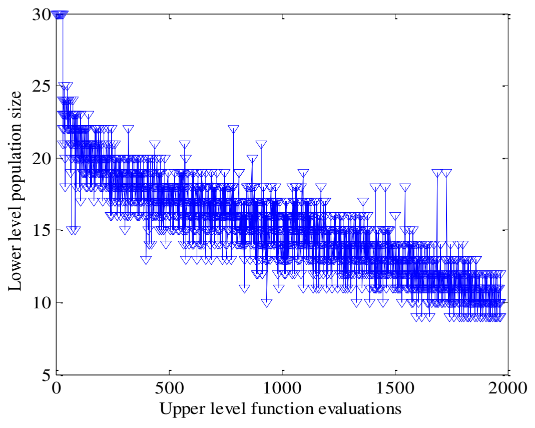
Table 4 CPU running time comparison among SABiLDE, SABLA, BL-CMA-ES, BLEAQ and BlDE

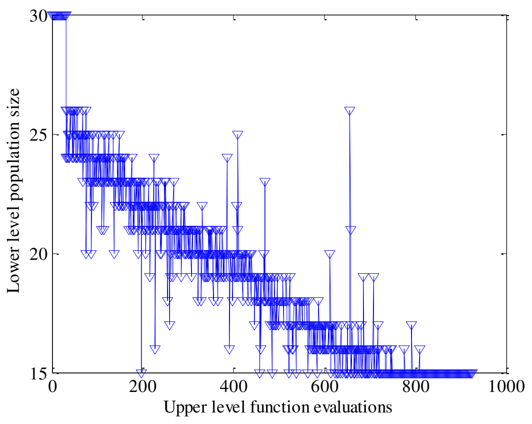
Prob	BLDE			SABiLDE			BLEAQ		
	mean	min	max	mean	min	max	mean	min	max
tp1	32.714	18.321	112.173	16.660	14.432	21.374	**6.678**	**4.559**	**8.646**
tp2	75.793	15.593	162.251	11.920	6.015	17.063	**6.012**	**2.670**	**10.903**
tp3	268.432	46.109	364.649	42.372	35.825	57.881	**10.570**	**8.632**	**11.317**
tp4	48.907	39.020	55.942	31.98	26.370	37.441	**9.099**	**5.947**	**10.823**
tp5	228.538	208.604	238.950	14.915	13.325	18.093	**8.067**	**4.883**	**11.640**
tp6	173.114	82.000	189.360	32.147	24.904	38.386	**9.721**	**7.216**	**13.644**
tp7	126.857	45.708	181.652	31.831	19.007	37.091	**18.101**	**13.423**	**24.456**
tp8	83.928	17.009	175.825	13.591	10.941	18.073	**5.413**	**3.118**	**10.586**
tp9	146.917	88.985	181.029	**12.742**	**9.731**	**15.730**	17.114	14.933	20.581
tp10	144.851	90.902	199.175	**12.371**	**10.930**	**14.405**	18.856	15.272	25.952
smd1	158.597	147.217	186.586	**44.080**	39.438	**48.787**	47.068	**28.369**	70.9977
smd2	351.467	148.855	526.594	**39.245**	35.063	**42.905**	42.716	**31.783**	55.2662
smd3	180.742	145.235	437.828	**36.595**	**33.143**	**40.650**	54.786	46.503	65.9844
smd4	489.516	147.623	526.934	**37.701**	32.538	**47.615**	51.386	**30.914**	72.4128
smd5	703.625	289.160	1068	66.735	52.587	77.819	**50.104**	**37.029**	**61.399**
smd6	854.257	840.717	886.098	318.958	116.008	583.446	**29.052**	**18.439**	**36.113**
smd7	484.555	448.323	539.513	113.226	79.106	158.942	**73.413**	**44.964**	**116.580**
smd8	473.222	244.372	2228	**56.939**	**48.455**	**67.575**	221.611	94.867	263.695
smd9	794.143	737.537	1181	**94.442**	74.411	**67.575**	244.192	**60.090**	313.705
smd10	1106.593	1030.802	1156	81.831	47.391	**109.026**	**53.790**	**32.236**	125.445
smd11	1209.654	569.861	1433	**56.558**	**37.672**	**77.921**	206.845	59.363	360.439
smd12	962.659	939.063	987.581	**41.754**	**30.930**	**47.643**	91.596	45.870	160.476



(a) TP test problems (b) SMD test problems

Fig. 4 The comparison of mean of CPU running time





(a) TP1 test problem   (b) SMD1 test problem

Fig.5 Lower level population size dynamics with upper level evolution

Finally, to achieve a closer inspection, the average CPU running time results of the Wilcoxon signed-rank statistic for all the test problems are shown in Table 5, from which it can be seen that SABiLDE is significantly better than BlDE, and competitive to BLEAQ at the two confidence levels of α = 0.1 and α = 0.05. Figure 6 presents the average Friedman test rankings of the three algorithms for all the test functions. These statistics further show that SABiLDE has better performance than BlDE. BLEAQ obtained a better rank, but the difference between SABiLDE and BLEAQ is not obvious. All these show that SABiLDE has better computational efficiency than BlDE, and competitive computational cost to BLEAQ.

Table 5 Wilcoxon test results of the computational efficiency among different BLEAs

	(1)	(2)	(3)
BlDE(1)	-	0.0 o	0.0 o
SABiLDE (2)	253.0 ●	-	110.0
BLEAQ (3)	253.0 ●	143.0	-

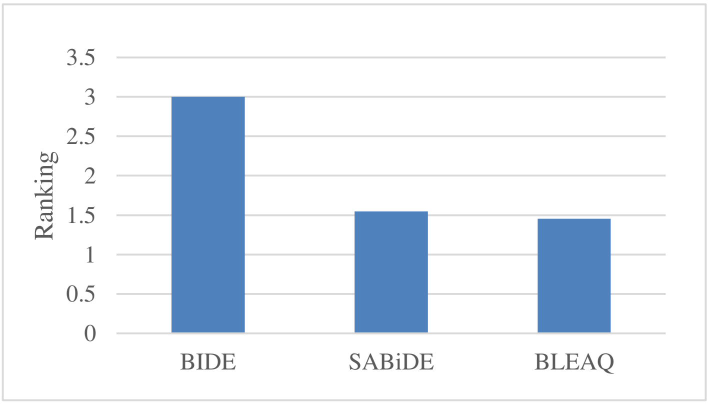


Fig.6 Average Friedman rankings of computational efficiency for different BLEAs

4.4 Evaluation of the proposed self-adaptive strategies

To demonstrate the sensitivity of the proposed self-adaptive strategies to the population size, search bound and mutation operator of lower level DE, four different variants of SABiLDE, without using the proposed self-adaptive strategies, were evaluated. The four variants of SABiLDE are as follows:

SABiLDE-I: without self-adaptively adjusting the lower level population size

SABiLDE-II: without self-adaptively adjusting the lower level search radius

SABiLDE-III: without self-adaptively adjusting the lower level population size and search radius

SABiLDE-IV: without self-adaptively adjusting the lower level mutation strategy

The upper and lower median error values of these SABiLDEs for the TP test problems are shown in Table 6. The upper and lower success rates of the five algorithms in 30 runs for the TP test set are also shown in Table 6. To evaluate the computational efficiency, the CPU running time of the five algorithms is recorded in Table 7. In order to highlight the overall best results, the significantly better values are marked in bold.

Table 6 Median solution accuracy comparison among SABiLDEs

Prob	SABiLDE			SABiLDE-I			SABiLDE-II			SABiLDE-III			SABiLDE-IV		
	UL Acc.	LL Acc.	SR(%)	UL Acc.	LL Acc.	SR(%)	UL Acc.	LL Acc.	SR(%)	UL Acc.	LL Acc.	SR(%)	UL Acc.	LL Acc.	SR(%)
tp1	8.85E-4	1.17E-3	100	5.13E-4	9.62E-3	100	5.82E-3	8.42E-3	100	**1.51E-4**	**9.42E-3**	100	1.72E-3	7.24E-2	100
tp2	6.42E-5	1.83E+0	93.3	4.57E-5	2.07E+1	86.7	5.77E-5	1.98E+1	86.7	4.91E-5	1.23E+1	90	**4.45E-5**	**8.62E-4**	100
tp3	**2.89E-5**	**9.34E-5**	100	1.80E-4	1.62E-4	100	2.61E-4	1.96E-4	100	1.45E-4	2.79E-3	100	1.80E-4	9.86E-5	100
tp4	**3.93E-4**	1.29E-6	100	3.89E-4	6.50E-4	100	4.71E-4	6.99E-4	100	6.05E-4	**1.00E-6**	100	6.16E-3	4.37E-4	100
tp5	3.65E-6	1.92E-6	100	3.10E-6	6.07E-6	100	3.41E-6	1.76E-6	100	**1.05E-6**	**1.18E-6**	100	2.21E-5	2.47E-6	100
tp6	7.77E-4	2.78E-3	100	7.77E-4	2.78E-3	100	7.80E-4	2.79E-3	100	7.79E-4	2.78E-3	100	7.77E-4	2.79E-3	100
tp7	8.01E-4	6.92E-4	100	7.96E-4	**6.11E-4**	100	8.10E-4	7.27E-4	100	**7.54E-4**	6.45E-4	100	7.92E-4	7.43E-4	100
tp8	**1.00E-6**	**1.00E-6**	100	2.24E-6	1.00E+1	93.3	2.45E-6	2.00E+1	86.7	6.84E-6	1.00E+1	96.6	2.21E-6	2.00E+1	86.7
tp9	**1.00E-6**	**1.00E-6**	100	**1.00E-6**	**1.00E-6**	100	**1.00E-6**	**1.00E-6**	100	**1.00E-6**	**1.00E-6**	100	**1.00E-6**	**1.00E-6**	100
tp10	2.58E-6	**1.00E-6**	100	2.82E-5	**1.00E-6**	100	2.73E-5	**1.00E-6**	100	**2.28E-5**	**1.00E-6**	100	2.39E-5	**1.00E-6**	100

Table 7 The CPU running time comparison among SABiLDEs

Prob.	SABiLDE		SABiLDE-I		SABiLDE-II		SABiLDE-III		SABiLDE-IV	
	mean	min	mean	min	mean	min	mean	min	mean	min
tp1	**16.66**	**14.432**	45.44	42.68	22.02	19.82	43.57	42.97	23.47	20.77
tp2	**11.92**	**6.015**	36.67	29.37	18.01	17.39	28.99	28.36	18.34	17.51
tp3	**42.37**	**35.825**	102.46	98.33	49.37	40.16	76.27	75.38	53.95	48.78
tp4	31.98	**26.370**	61.99	54.76	**31.65**	27.57	52.84	50.82	49.32	46.13
tp5	**14.92**	**13.325**	41.96	39.43	20.33	18.68	132.75	88.53	33.47	31.88
tp6	**32.15**	**24.904**	86.70	71.87	35.50	30.20	176.70	97.39	61.28	59.49
tp7	**31.83**	**19.007**	71.81	67.44	35.52	29.41	71.26	69.36	48.17	44.34
tp8	**13.59**	**10.941**	32.94	30.12	16.57	12.39	30.13	28.45	22.91	20.74
tp9	**12.74**	**9.731**	38.58	34.54	22.86	16.64	39.55	37.27	29.28	25.33
tp10	**12.37**	**10.930**	32.63	31.05	22.58	16.78	39.57	38.16	30.42	25.06

From Table 6, the difference of upper and lower convergence accuracy among the SABiLDEs is not obvious, but SABiLDE-I is generally able to obtain better results than SABiLDE for most of the test problems. The reason is that the lower level population of SABiLDE-I is fixed on the initial population size, enabling it to always have a larger lower level population size than SABiLDE. It is well known that, for the population based stochastic optimization algorithms, a larger population size is generally able to obtain a better result for the same stopping criteria. However, the cost is to perform more function evaluations. From Table 7, it can be seen that SABiLDE-I costs more CPU running time than SABiLDE for all the test problems.

SABiLDE has better solution accuracy than that of SABiLDE-II with fix search bounds, because a smaller search radius is desirable for the LLDE to do local search and the solution accuracy and search efficiency can thus be enhanced. Accordingly, it can be observed from Table 7 that SABiLDE generally has less CPU running time than SABiLDE-II.

When no self-adaption is introduced for either the lower level population size and search radius, it is expected that SABiLDE-III has good convergence performance but bad computation efficiency. It can be observed from Table 6 that SABiLDE-III performs better than SABiLDE on the upper problems of TP2, TP5, TP7, TP10 and lower problems of TP1, TP4, TP5, TP7, TP10. But when it comes to the computation efficiency, as shown in Table 7, SABiLDE-III got the worst result compared with the other four SABiLDEs.

As for SABiLDE-IV, because of the use of the DE/target-to-best/1 mutation strategy, a good balance between the exploration and exploitation can be achieved. It can be seen from Table 6 that SABiLDE-IV converges to the optimal solutions of all the test problems with success rate of 100%. For SABiLDE, the DE/best/1 is able to adaptively do local search to speed up the convergence. As shown in Table 7, SABiLDE has faster convergence speed than SABiLDE-IV.

The above test results show that the proposed self-adaptive strategies are beneficial to improve the computational efficiency of the lower level DE, therefore, the computation efficiency of the whole bilevel DE is enhanced.

5. Conclusions

The paper introduced a new method for effectively solving bilevel optimization problems. The scheme is to efficiently find solutions for the lower level task, which produce good responses to the upper level task; in this way, the overall efficiency of solving bilevel optimization is significantly improved. Specifically, a novel efficient self-adaptive bilevel differential evolution (SABiLDE) with *k*-NN approximation for the lower level optimization was proposed. As detailed in Section 3, the proposed the self-adaptive control rate, together with the introduction of the archiving technique, plays a key role in effectively improving the overall efficiency of the optimization algorithm.

The archiving technique is used to store all the feasible lower level solutions obtained by the lower level DE during the whole evolution and the corresponding upper decision variables. Based on the paired upper and lower solutions in the archive, *k*-nearest neighbors are identified for a newly generated upper candidate to approximate the optimal lower level variables by the inverse distance weighting interpolating. If the distance between the new upper candidate and its closest archived member is small enough, the approximated lower level variables are directly accepted as the optimal lower variables and the lower level DE does not need to perform. Otherwise, the approximated lower solution is used as a base individual to generate an initial population for the lower DE to speed up the convergence rate, because the surrogate model may not be an excellent approximation of the bilevel problem, but it is usually still a good prediction and can provide useful information to direct the search in some better regions. Based on the similarity (the nearest neighbor distance), the self-adaptive control rate is proposed to dynamically adjust the lower level population size and search radius to reduce the lower level function evaluations and therefore to improve the computational efficiency.

The performance of the proposed algorithm was evaluated on a test set with 10 standard bilevel test problems and the SMD benchmark suite with 12 scalable test problems. The test results show that the proposed SABiLDE is an efficient and effective approach for generic bilevel optimization problems. Compared with another *k*-NN assisted bilevel differential evolution, called BlDE, the proposed algorithm displayed better performance in both convergence accuracy and computational efficiency for almost all the test problems. Compared with the modified BLEAQ, SABiLDE is able to provide better or competitive results, especially on the test problems with high dimensions. In comparison with two of the lately developed BLEAs, namely, SABLA and BL-CMA-ES, SABiLDE also demonstrates competitive convergence performance.

For the future work, we are planning to apply the proposed algorithm to the bilevel robust dynamic economic emission dispatch of power systems. Moreover, it is also of our interest to investigate more efficient surrogate models such as radial base functions to approximate the lower level variables in future.

Acknowledgment: This work was partly supported by the National Natural Science Foundation of P. R. China (Grant no. 61203309), Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/I011056/1 and Platform Grant EP/H00453X/1, and Natural and Environment Research Council (NERC) under the grant NE-V002511, National Defense Basic Research Program of China (Grant no. JCKY2019403D006), Hunan Provincial Natural Science Foundation of China (Grant No. 2018JJ2137), Hunan Provincial Science and Technology Plan of China (Grant no. 2017XK2302) and Hunan Provincial Innovation Foundation for Postgraduate (CX20190807).

References:

[1] Benth F E, Dahl G, and Mannino C (2012) Computing optimal recovery policies for financial markets. Oper Res 60(6):1373-1388

[2] Chiou S W (2009) A bi-level programming for logistics network design with system-optimized flows. Inf Sci 179: 2434-2441

[3] Zhang G, Gao Y, Lu J (2011) Competitive strategic bidding optimization in electricity markets using bilevel programming and swarm technique. IEEE Trans Ind Electron 58:2138-2146

[4] Calvete H I, Galé C, Oliveros M J (2011) Bilevel model for production distribution planning solved by using ant colony optimization. Comput Oper Res 38:320-327

[5] Kuo R J, Han Y S (2011) A hybrid of genetic algorithm and particle swarm optimization for solving bi-level linear programming problem – a case study on supply chain model. Applied Mathematical Modelling 35:3905-3917.

[6] Koh A (2007) Solving transportation bi-level programs with differential evolution. In IEEE Congress on Evolutionary Computation. IEEE, pp. 2243-2250

[7] Sinha A, Malo P, and Deb K (2015) Transportation policy formulation as a multi-objective bilevel optimization problem. In 2015 IEEE Congress on Evolutionary Computation (CEC-2015).

[8] Wein L (2009) Homeland security: from mathematical models to policy implementation: the 2008 Philip McCord Morse lecture. Oper Res 57(4):801-811

[9] Shabde V S, Hoo K A (2008) Optimum controller design for a spray drying process. Control Engineering Practice 16:541-552

[10] Lu J, Han J, Hu Y, and Zhang G (2016) Multilevel decision-making: A survey. Inf Sci 346-347:463-487

[11] Sinha A, Malo P, and Deb K (2013) Efficient evolutionary algorithm for single-objective bilevel optimization. CoRR, abs/1303.3901.

[12] Sinha A, Malo P, Deb K, Korhonen P and Wallenius J (2016) Solving bilevel multi-criterion optimization problems with lower level decision uncertainty. IEEE Trans Evol Comput 20(2):199-217

[13] Hansen P, Jaumard B, and Savard G (1992) New branch-and-bound rules for linear bilevel programming. SIAM J Sci and Statis Comput 13(5):1194-1217.

[14] Sinha A, Malo P, and Deb K (2014) An improved bilevel evolutionary algorithm based on quadratic approximations. In 2014 IEEE Congress on Evolutionary Computation (CEC-2014). IEEE, pp. 1870-1877

[15] Colson B, Marcotte P, Savard G (2007) An overview of bilevel optimization. Ann Oper Res 153: 235-256.

[16] Storn R and Price K (1977) Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341-359

[17] Angelo J S, Krempser E, Barbosa H J C (2014) Differential evolution assisted by a surrogate model for bilevel programming problems. In 2014 IEEE Congress on Evolutionary Computation (CEC-2014). pp. 1784-1791

[18] Sinha A, Malo P, and Deb K (2012) Unconstrained scalable test problems for single-objective bilevel optimization. In 2012 IEEE World Congress on Computational Intelligence, 2012.

[19] Sinha A, Malo P, and Deb K (2014) Test problem construction for single-objective bilevel optimization. Evol Comput 22(3):439-477

[20] Deb K and Sinha A (2010) An efficient and accurate solution methodology for bilevel multi-objective programming problems using a hybrid evolutionary-local-search algorithm. Evol Comput 18(3):403-449.

[21] Sinha A, Malo P, and Deb K (2015) Towards understanding bilevel multi-objective optimization with deterministic lower level decisions. In Proceedings of the Eighth International Conference on Evolutionary Multi-Criterion Optimization (EMO-2015). Springer-Verlag, 2015.

[22] Hejazi S, Memariani A, Jahanshahloo G, and Sepehri M (2002) Linear bilevel programming solution by genetic algorithm. Comput & Oper Res 29(13):1913-1925

[23] Wan Z, Wang G, Sun B (2013) A hybrid intelligent algorithm by combining particle swarm optimization with chaos searching technique for solving nonlinear bilevel programming problems. Swarm and Evol Comput 8:26-32.

[24] Z. Wan, L. Mao, G. Wang. (2014) Estimation of distribution algorithm for a class of nonlinear bilevel programming problems. Inf Sci 256:184-196.

[25] Wang Y, Jiao Y C, and Li H (2005) An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme. IEEE Trans Sys Man and Cyber Part C: Appl and Reviews 35(2):221-232

[26] Jiang Y, Li X, Huang C, Wu X (2013) Application of particle swarm optimization based on CHKS smoothing function for solving nonlinear bilevel programming problem. Appl Math Comput 219:4332-4339

[27] Li H (2015) A genetic algorithm using a finite search space for solving nonlinear/linear fractional bilevel programming problems. Ann Oper Res 235:543-558

[28] Mathieu R, Pittard L, and Anandalingam G (1994) Genetic algorithm based approach to bi-level linear programming. Oper Res 28(1):1-21

[29] Yin Y (2000) Genetic algorithm based approach for bilevel programming models. J of Transport Eng, 126(2):115-120

[30] Zhu X, Yu Q, and Wang X (2006) A hybrid differential evolution algorithm for solving nonlinear bilevel programming with linear constraints. In the 5th IEEE International Conference on Cognitive Informatics. IEEE, pp. 126-131

[31] Koh A (2007) Solving transportation bi-level programs with differential evolution. In IEEE Congress on Evol Comput. IEEE, pp. 2243-2250

[32] Islam M M, Singh H K and Ray T (2015) A memetic algorithm for solving single objective bilevel optimization problems. In 2015 IEEE Congress on Evolutionary Computation (CEC-2015). IEEE, pp. 1643-1650

[33] Gao Y, Zhang G, Lu J, Wee H M (2011) Particle swarm optimization for bi-level pricing problems in supply chains. J Global Optim 51:245-254

[34] Zhao L, Wei J X. (2019) A nested particle swarm algorithm based on sphere mutation to solve bi-level optimization. Soft Comput 23:11331-11341

[35] Sinha A, Malo P, Frantsev A, and Deb K (2014) Finding optimal strategies in a multi-period multi-leader-follower stackelberg game using an evolutionary algorithm. Comput Oper Res 41:374-385.

[36] Angelo J S, Krempser E, Barbosa H J C (2013) Differential evolution for bilevel programming. In 2013 IEEE Congress on Evolutionary Computation (CEC-2013). IEEE, pp. 470-477

[37] He X, Zhou Y, Chen Z. (2018) Evolutionary Bilevel Optimization based on Covariance Matrix Adaptation. IEEE Trans Evol Comput 23(2):258-272.

[38] Huang P Q, Wang Y (2020) A Framework for Scalable Bilevel Optimization: Identifying and Utilizing the Interactions Between Upper-Level and Lower-Level Variables. IEEE Trans Evol Comput 24(6):1150-1163

[39] Oduguwa V and Roy R (2002) Bi-level optimization using genetic algorithm. In Proceedings of the 2002 IEEE International Conference on Artificial Intelligence Systems. IEEE, pp.123-128

[40] Legillon F, Liefooghe A, and Talbi E G (2012) Cobra: a cooperative coevolutionary algorithm for bi-level optimization. In 2012 IEEE Congress on Evolutionary Computation (CEC-2012). IEEE, 2012.

[41] Chaabani A, Bechikh S, Said L B (2015) A co-evolutionary decomposition-based algorithm for bi-level combinatorial optimization. In IEEE Congress on Evolutionary Computation. IEEE, pp. 1659-1666

[42] Chaabani A, Bechikh S, Said L B (2018) A co-evolutionary hybrid decomposition-based algorithm for bi-level combinatorial optimization problems. Appl Intelligence 48:2847-2872

[43] Li H, Fang L (2014) Co-evolutionary algorithm: an efficient approach for bilevel programming problem. Eng Optim 46(3):361-374.

[44] Said R, Elarbi M, Bechikh S, Said L B (2021) Solving combinatorial bi-level optimization problems using multiple populations and migration schemes. Oper Res doi.org/10.1007/s12351-020-00616-z.

[45] Sinha A, Lu Z, Deb K, Malo P (2020) Bilevel optimization based on iterative approximation of multiple mappings. J Heuristics 26:151-185

[46] Islam M, Singh H K, Ray T (2017) A Surrogate Assisted Approach for Single-Objective Bilevel Optimization. IEEE Trans Evol Comput 21(5):681-696

[47] Singh H K, Islam M, Ray T, Ryan M J (2019) Nested evolutionary algorithms for computationally expensive bilevel optimization problems: Variants and their systematic analysis. Swarm Evol Comput 48:329-344

[48] Shepard D (1968) A two-dimensional interpolation function for irregularly-spaced data. In Proc. of the 23rd ACM National Conference. ACM, pp. 517-524.

[49] Das S and Suganthan P N (2011) Differential evolution: A survey of the state-of-the-art. IEEE Trans Evol Comput 15(1): 4-31

[50] Qin A K, Huang V L, and Sugannthan P N (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2): 398-417

[51] Mezura-Montes E, Velázquez-Reyes J, and Coello Coello C A (2006) A comparative study of

differential evolution variants for global optimization. In Proc. Genet. Evol. Comput. Conf. pp. 485-492.

[52] Derrac J, GarcíaS, Molina D (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evolut Comput 1(1):3-18

[53] Alcalá-Fdez J, Sánchez L, García S et al (2009) KEEL: a software tool to assess evolutionary algorithms for data mining problems. Soft Comput 13(3):307-318