Chinese Journal of Electronics
Vol.XX, No.X, Jan. 20XX

Principled Design of Translation, Scale, and Rotation Invariant Variation Operators for Metaheuristics

TIAN Ye[1], ZHANG Xingyi[2], HE Cheng[3], TAN Kay Chen[4] and JIN Yaochu[5]

(1. *Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, Institutes of Physical Science and Information Technology, Anhui University, Hefei 230601, China*)

(2. *School of Computer Science and Technology, Anhui University, Hefei 230601, China*)

(3. *School of Electrical and Electronic Engineering, Huazhong University of Science and Technology, Wuhan 430074, China*)

(4. *Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR, China*)

(5. *Faculty of Technology, Bielefeld University, 33619 Bielefeld, Germany*)

Abstract — In the past three decades, a large number of metaheuristics have been proposed and shown high performance in solving complex optimization problems. While most variation operators in existing metaheuristics are empirically designed, this paper aims to design new operators automatically, which are expected to be search space independent and thus exhibit robust performance on different problems. For this purpose, this work first investigates the influence of translation invariance, scale invariance, and rotation invariance on the search behavior and performance of some representative operators. We deduce the generic form of translation, scale, and rotation invariant operators. A principled approach is proposed for the automated design of operators, which searches for high-performance operators based on the deduced generic form. The experimental results demonstrate that the operators generated by the proposed approach outperform state-of-the-art ones on a variety of problems with complex landscapes and up to 1000 decision variables.

Key words — Metaheuristics, Evolutionary computation, Swarm intelligence, Variation operator, Transformation invariance, Automated design.

1 Introduction

Metaheuristics have shown effectiveness in solving the complex optimization problems from various fields, such as manufacturing[1], scheduling[2], bioinformatics[3], and economics[4]. In contrast to mathematical programming methods that require an explicit objective function[5], metaheuristics provide a high-level methodology solving problems in a black-box manner. So far, there have been a variety of algorithms inspired by the biological mechanisms in natural evolution, such as genetic algorithms (GA)[6], differential evolution (DE)[7], evolution strategies[8], and evolutionary programming[9]. Moreover, many swarm intelligence based algorithms have also been proposed, including particle swarm optimization (PSO)[10], ant colony optimization[11], artificial bee colony algorithms[12], among many others.

Existing metaheuristics exhibit quite different search behaviors and optimization performance, which are mainly determined by the manually designed variation operators[13, 14, 15], i.e., the strategies for receiving the decision vectors of parents and generating the decision vectors of offspring solutions. For example, the GA was designed according to the evolution theory and law of inheritance, which generates offspring by the crossover between two parents and the mutation on a single offspring solution. The crossover and mutation operators provide a powerful exploration ability[16], making GA good at han-

*Manuscript Received June XX, 20XX; Accepted July XX, 20XX. This work was supported in part by the National Key R&D Program of China under Grant 2018AAA0100100, in part by the National Natural Science Foundation of China under Grant 61876162, Grant 61906001, Grant 61903178, Grant 62136008, Grant U20A20306, and Grant U21A20512, and in part by an Alexander von Humboldt Professorship for Artificial Intelligence funded by the Federal Ministry of Education and Research, Germany. (Corresponding author: Xingyi Zhang.)

 DOI:10.1049/cje.20XX.0X.0XX

dling multimodal landscapes[17]. The DE mutates each solution according to the weighted difference between the other two solutions, which holds a good performance on problems with complicated variable linkages[18]. The covariance matrix adaptation evolution strategy (CMA-ES) generates new solutions by sampling a multivariate normal distribution model adaptively learned from the population, showing high performance on many real-world applications[19]. Inspired by the choreography of bird flocking, the PSO updates each particle according to its personal best particle and the global best particle, which has a high speed of convergence[20].

Among the superiorities in existing variation operators, the independence of search space is crucial to the robustness and generalization of metaheuristics, since metaheuristics do not rely on specific characteristics of problems. A search space independent operator holds the same performance on a problem with arbitrary search space transformations, including translation (i.e., $x' = x + b$), scaling (i.e., $x' = ax$), and rotation (i.e., $\mathbf{x}' = \mathbf{x}M$). Existing studies have shed some light on the sufficient conditions of achieving these invariance properties. For instance, CMA-ES is scale invariant since its step-sizes are set proportionally to the distance to the optimum found so far[21], and the mutation operator of DE is rotation invariant since it is the weighted sum of multiple parents[22]. Nevertheless, the mathematical definitions of all the three properties have not been given, and the sufficient and necessary condition of achieving them is still unknown. Therefore, it is difficult to analyze whether an operator is translation, scale, and rotation invariant theoretically, or to consider these properties in designing new operators explicitly.

To address this issue, this work aims to deduce the generic form (i.e., sufficient and necessary condition) of translation, scale, and rotation invariant operators, which can be used to judge the possession of invariance properties and guide the design of new operators. Based on the deduced generic form, this work proposes a principled approach for designing new operators with invariance properties. In contrast to the parameter tuning of metaheuristics[23, 24], the off-line recommendation of metaheuristics[25, 26], and the on-line combination of metaheuristics[27, 28], the proposed approach is not based on any existing metaheuristic but searches for totally new operators. Moreover, the proposed approach does not utilize any existing optimizer or classifier (e.g., F-Race for parameter tuning[29], artificial neural network for metaheuristic recommendation[30], and sum-of-ranks multiarmed bandit algorithm for metaheuristic combination[31]). By contrast, it is a self-contained approach that can search for new operators by itself. The main components of this work include the following three aspects:

- **Theoretical analysis:** To illustrate the importance of translation invariance, scale invariance, and rotation invariance, their effects on the search behavior and performance of metaheuristics are investigated. Then, the sufficient and necessary condition of achieving these properties is mathematically derived, which reveals the generic form of search space independent operators.

- **New approach:** A principled approach to automated design of variation operators is proposed, termed AutoV. Based on the deduced generic form of variation operators, AutoV converts the search of high-performance operators into an optimization problem, in which the decision variables are the parameters in the operators. This way, AutoV can solve optimization problems without relying on any existing operators.

- **Experimental study:** The variation operator found by AutoV is embedded in a simple evolutionary framework and compared with eight classical or state-of-the-art metaheuristics. The experimental results show that the operator found by AutoV can obtain the best results on various challenging benchmark problems; in particular, it outperforms the winner of the CEC competition that contains multiple operators with complex adaptation strategies. The results indicate that AutoV has the potential to replace the laborious process of manual design of new metaheuristics.

The rest of this paper is organized as follows. Section 2 analyzes the effects of the three invariance properties, and Section 3 deduces their sufficient and necessary condition. Section 4 presents the proposed principled approach, and Section 5 gives the experimental studies. Finally, Section 6 concludes this paper.

2 Effects of Invariance Properties

This work focuses on the variation operators for the following continuous optimization problem:

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \end{aligned}, \tag{1}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_D)$ is a decision vector denoting a solution for the problem, $\mathbf{l} = (l_1, l_2, \ldots, l_D)$ denotes the lower bound, $\mathbf{u} = (u_1, u_2, \ldots, u_D)$ denotes the upper bound, and D is the number of decision variables. To solve such problems in a black-box manner, a variety of variation operators have been proposed to generate solutions without using any specific information except for

the lower and upper bounds. This section first introduces some representative variation operators, then presents the definitions of translation invariance, scale invariance, and rotation invariance and analyzes their effects by examples and experimental studies.

2.1 Variation Operators

An operator generally receives the decision vectors of one or more parents, then outputs the decision vectors of one or more offspring solutions. For example, the simulated binary crossover (SBX) operator[32] used in GA uses two parents $\mathbf{x}_1$ and $\mathbf{x}_2$ to generate two offspring solutions $\mathbf{o}_1$ and $\mathbf{o}_2$ each time:

$$\begin{cases} o_{1d} = 0.5\left[(1+\beta)x_{1d} + (1-\beta)x_{2d}\right] \\ o_{2d} = 0.5\left[(1-\beta)x_{1d} + (1+\beta)x_{2d}\right] \end{cases}, \ 1 \le d \le D \ , \quad (2)$$

where o_{1d} denotes the d-th variable of solution $\mathbf{o}_1$ and β is a random number obeying a special distribution. The mutation operator of DE/rand/1/bin[7] uses three parents $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ to generate an offspring solution $\mathbf{o}$ each time:

$$o_d = x_{1d} + F \cdot (x_{2d} - x_{3d}) \ , \ 1 \le d \le D \ , \quad (3)$$

where F is a parameter controlling the amplification of the difference between $\mathbf{x}_2$ and $\mathbf{x}_3$. In contrast to the random parameter β in SBX that varies on each dimension, the parameter F in DE is a predefined constant. The operator of CMA-ES[8] generates offspring by sampling a multi-variate normal distribution:

$$\mathbf{o} = \mathbf{x}_m + \sigma \cdot \mathcal{N}(\mathbf{0}, C) \ , \quad (4)$$

where $\mathbf{x}_m$ is the weighted sum of all solutions, σ is a vector of iteratively updated step-sizes, and C is a covariance matrix updated according to the current population. In general, the initial σ can be set to $0.6(\mathbf{u}-\mathbf{l})$[33]. Similar to CMA-ES, the operator of fast evolutionary programming (FEP)[9] generates offspring by sampling a single-variate normal distribution:

$$o_d = x_d + \eta_d \cdot \mathcal{N}(0,1) \ , \ 1 \le d \le D \ , \quad (5)$$

where η is a vector of self-adaptive standard deviations related to each solution $\mathbf{x}$, whose elements can be initialized to 3[9].

It can be found that these operators generate offspring by using distinct formulas. Generally, an operator can be regarded as a function $h(x_{1d}, x_{2d}, \ldots)$ performed on each dimension d, where $x_{1d}, x_{2d}, \ldots$ contains the decision variables of parents, the lower bound, and the upper bound. Note that other parameters (e.g., β in SBX and C in CMA-ES) are ignored for simplicity. In the following, we investigate how the invariance properties influence the search behavior and performance of operators.

2.2 Effects of Translation Invariance

According to Ref.[34], a variation operator $h(x_{1d}, x_{2d}, \ldots)$ is invariant to search space transformation $\mathcal{T}$ means that $h(\mathcal{T}(x_{1d}), \mathcal{T}(x_{2d}), \ldots) = \mathcal{T}(h(x_{1d}, x_{2d}, \ldots))$. A translation of the search space can be regarded as an addition of each decision variable, i.e., $\mathcal{T}(x) = x + b$, hence the translation invariance property can be defined as follows:

Definition 1 (**Translation invariance**). *A variation operator $h(x_{1d}, x_{2d}, \ldots)$ is translation invariant if and only if*

$$h(x_{1d} + b, x_{2d} + b, \ldots) = h(x_{1d}, x_{2d}, \ldots) + b \quad (6)$$

holds for any real constant b.

It is not difficult to find that all the operators of SBX, DE, CMA-ES, and FEP are translation invariant. In particular, the SBX operator described in (2) can be rewritten as

$$h_{sbx}(x_{1d}, x_{2d}) = x_{1d} + 0.5(1 \pm \beta)(x_{2d} - x_{1d}) \ , \quad (7)$$

hence

$$\begin{aligned} &h_{sbx}(x_{1d} + b, x_{2d} + b) \\ &\quad = x_{1d} + b + 0.5(1 \pm \beta)(x_{2d} + b - x_{1d} - b) \\ &\quad = h_{sbx}(x_{1d}, x_{2d}) + b \end{aligned} \quad (8)$$

and the operator is translation invariant. By contrast, if the SBX operator is modified to

$$h_{sbx'}(x_{1d}, x_{2d}) = 0.1x_{1d} + 0.5(1 \pm \beta)(x_{2d} - x_{1d}) \ , \quad (9)$$

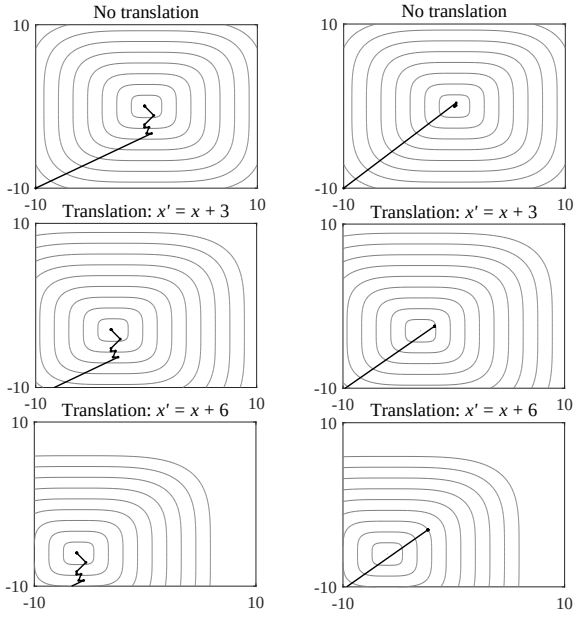
then

$$\begin{aligned} &h_{sbx'}(x_{1d} + b, x_{2d} + b) \\ &\quad = 0.1x_{1d} + 0.1b + 0.5(1 \pm \beta)(x_{2d} + b - x_{1d} - b) \\ &\quad \neq h_{sbx'}(x_{1d}, x_{2d}) + b \end{aligned} \quad (10)$$

and the operator becomes not translation invariant. Obviously, the modified SBX′ operator is likely to evolve the population toward the origin.

To better illustrate this fact, Fig. 1 depicts the convergence profiles of SBX and SBX′ based GAs on problem $f(x_1, x_2) = x_1^4 + x_2^4$ with and without translation, where the random number seed is fixed and both the population size and the number of generations are set to 10. It can be found that the SBX based GA holds the same search behavior and can always converge to the global optimums of the three problems, whereas the SBX′ based GA always converges to the origin. In short, the SBX operator is translation invariant but SBX′ is not. Furthermore, Table 1 lists the mean and standard deviation of the minimum objective values obtained by SBX and SBX′ based GAs on six benchmark problems[17], averaged over 30 runs. The six problems have the same global optimum $(0, 0, \ldots, 0)$, while the global optimum is

changed to $(-6,-6,\ldots,-6)$ if the problems are translated by $x' = x + 6$. For the original problems, the SBX′ based GA significantly outperforms the SBX based GA since the former can quickly converge to the origin. While for the translated problems, the performance of SBX′ based GA deteriorates considerably since it cannot converge to the translated global optimum; by contrast, the performance of SBX based GA keeps unchanged since the SBX operator is translation invariant.



(a) SBX based GA (translation invariant) (b) SBX′ based GA

Fig. 1: Convergence profiles of SBX and SBX′ based GAs with fixed random number seed on $f(x_1, x_2) = x_1^4 + x_2^4$ with and without translation

Table 1: Minimum objective values obtained by SBX and SBX′ based GAs on six problems with and without translation

Original problem $(x_1,\ldots,x_{30} \in [-10,10])$	SBX based GA (translation invariant)	SBX′ based GA
Schwefel's Function 2.22	5.4310e+0 (2.13e+0)	1.0965e-49 (1.29e-49)
Schwefel's Function 2.21	4.5792e+0 (1.21e+0)	8.3896e-49 (1.23e-48)
Quaric Function	1.6425e+3 (1.03e+3)	2.9477e-4 (2.66e-4)
Griewank Function	4.7700e-1 (1.87e-1)	0.0000e+0 (0.00e+0)
Ackley's Function	3.3728e+0 (3.75e-1)	8.8818e-16 (0.00e+0)
Rastrigin's Function	4.2269e+1 (1.08e+1)	0.0000e+0 (0.00e+0)
Translated problem $(x' = x + 6)$	SBX based GA (translation invariant)	SBX′ based GA
Schwefel's Function 2.22	5.2237e+0 (1.88e+0)	1.0083e+12 (2.18e+12)
Schwefel's Function 2.21	4.9401e+0 (1.24e+0)	5.0000e+0 (0.00e+0)
Quaric Function	1.9875e+3 (1.62e+3)	2.3718e+5 (1.53e+4)
Griewank Function	4.6346e-1 (1.17e-1)	1.1523e+0 (4.76e-3)
Ackley's Function	3.5154e+0 (7.18e-1)	1.2639e+1 (5.98e-4)
Rastrigin's Function	4.5695e+1 (1.01e+1)	7.4895e+2 (1.83e-1)

2.3 Effects of Scale Invariance

Similarly, a scaling of the search space can be regarded as a multiplication of each decision variable, i.e., $\mathcal{T}(x) = ax$, hence the scale invariance property can be defined as follows:

Definition 2 (**Scale invariance**). *A variation operator $h(x_{1d}, x_{2d}, \ldots)$ is scale invariant if and only if*

$$h(ax_{1d}, ax_{2d}, \ldots) = a \cdot h(x_{1d}, x_{2d}, \ldots) \tag{11}$$

holds for any real constant a.

It can be found that the operators of SBX, DE, and CMA-ES are scale invariant but the operator of FEP is not. Considering that the step-size σ is proportionally related to the decision space, the operator of CMA-ES described in (4) can be rewritten as

$$h_{cmaes}(x_{md}, l_d, u_d) = x_{md} + (u_d - l_d) \cdot \mathcal{N}(0, {\sigma'_d}^2 c) \ , \tag{12}$$

where σ'_d is a parameter within $(0, 1]$ and c is related to the covariance matrix C. Therefore,

$$\begin{aligned} &h_{cmaes}(ax_{md}, al_d, au_d) \\ &\quad = ax_{md} + (au_d - al_d) \cdot \mathcal{N}(0, {\sigma'_d}^2 c) \\ &\quad = a \cdot h_{cmaes}(x_{md}, l_d, u_d) \ , \end{aligned} \tag{13}$$

which means that the operator is scale invariant. On the other hand, the operator of FEP described in (5) can be rewritten as

$$h_{fep}(x_d) = x_d + \mathcal{N}(0, \eta_d^2) \ , \tag{14}$$

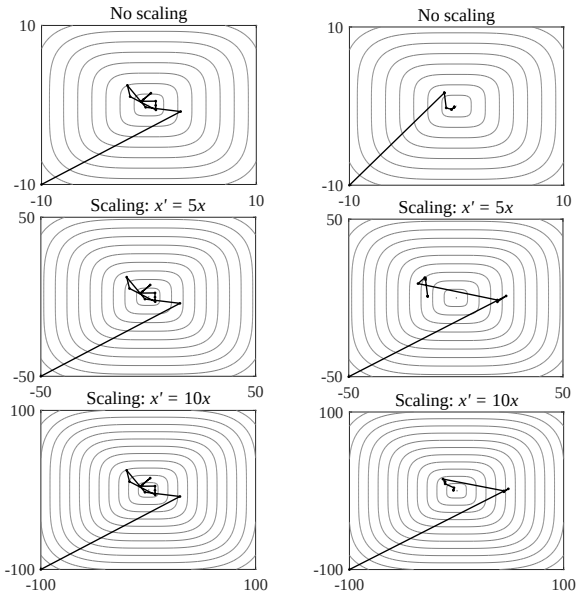
hence

$$h_{fep}(ax_d) = ax_d + \mathcal{N}(0, \eta_d^2) \neq a \cdot h_{fep}(x_d) \ , \tag{15}$$

which means that the operator is not scale invariant.

Fig. 2 plots the convergence profiles of CMA-ES and FEP on $f(x_1, x_2) = x_1^4 + x_2^4$ with and without scaling. It can be seen that the search behaviors of CMA-ES are the same on the three problems with different scales, whereas the search behaviors of FEP are quite different. Moreover, Table 2 presents the mean and standard deviation of the minimum objective values obtained by CMA-ES and FEP on six benchmark problems. Obviously, CMA-ES is competitive to FEP on the original problems, while CMA-ES dramatically outperforms FEP on the scaled problems. This is because the operator of CMA-ES is scale invariant and thus exhibits the same performance on a problem with different scales; by contrast, the operator of FEP is not scale invariant and thus is sensitive to the scales of

problems.



(a) CMA-ES (scale invariant) (b) FEP

Fig. 2: Convergence profiles of CMA-ES and FEP with fixed random number seed on $f(x_1, x_2) = x_1^4 + x_2^4$ with and without scaling

Table 2: Minimum objective values obtained by CMA-ES and FEP on six problems with and without scaling

Original problem ($x_1, \ldots, x_{30} \in [-10, 10]$)	CMA-ES (scale invariant)	FEP
Schwefel's Function 2.22	1.8226e+1 (3.38e+0)	1.6919e+1 (5.30e+0)
Schwefel's Function 2.21	1.0000e+1 (0.00e+0)	4.6087e+0 (2.29e-1)
Quaric Function	2.5024e+1 (9.92e+0)	1.4005e+3 (1.69e+3)
Griewank Function	3.2242e-1 (5.74e-2)	8.5491e-1 (1.18e-1)
Ackley's Function	2.7634e+0 (1.71e-1)	5.7111e+0 (5.97e-1)
Rastrigin's Function	2.1628e+2 (4.87e+0)	1.6904e+2 (2.52e+1)
Scaled problem ($x' = 10x$)	**CMA-ES (scale invariant)**	**FEP**
Schwefel's Function 2.22	1.9390e+1 (2.70e+0)	4.6714e+10 (5.47e+10)
Schwefel's Function 2.21	1.3107e+0 (1.36e-1)	8.4873e+0 (5.56e-1)
Quaric Function	2.6711e+1 (2.22e+1)	3.8536e+5 (2.35e+4)
Griewank Function	2.8246e-1 (9.96e-2)	1.1561e+0 (8.63e-3)
Ackley's Function	3.0355e+0 (3.42e-1)	1.3471e+1 (5.21e-1)
Rastrigin's Function	2.1651e+2 (1.40e+1)	9.2797e+2 (3.64e+1)

2.4 Effects of Rotation Invariance

A rotation of the search space can be regarded as a matrix multiplication of each decision vector, i.e., $\mathcal{T}(\mathbf{x}) = \mathbf{x}M$, hence the rotation invariance property can be defined as follows:

Definition 3 (**Rotation invariance**). *A variation operator $h(\mathbf{x}_1, \mathbf{x}_2, \ldots)$ is rotation invariant if and only if*

$$h(\mathbf{x}_1 M, \mathbf{x}_2 M, \ldots) = h(\mathbf{x}_1, \mathbf{x}_2, \ldots)M \tag{16}$$

holds for any orthogonal matrix M.

Note that here the decision variables on all dimensions $\mathbf{x}_1, \mathbf{x}_2, \ldots$ rather than those on a single dimension $x_{1d}, x_{2d}, \ldots$ should be considered. It can be deduced that the mutation operator of DE is rotation invariant while the operators of SBX, CMA-ES, and FEP are not. In particular, the mutation operator of DE described in (3) can be rewritten as

$$h_{de}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \mathbf{x}_1 + F \cdot (\mathbf{x}_2 - \mathbf{x}_3) \ , \tag{17}$$

hence

$$\begin{aligned} h_{de}(\mathbf{x}_1 M, \mathbf{x}_2 M, \mathbf{x}_3 M) &= \mathbf{x}_1 M + F \cdot (\mathbf{x}_2 M - \mathbf{x}_3 M) \\ &= h_{de}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)M \ , \end{aligned} \tag{18}$$

and the operator is rotation invariant. By contrast, since the SBX operator described in (7) can be rewritten as

$$h_{sbx}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 + (\mathbf{x}_2 - \mathbf{x}_1)B \ , \tag{19}$$

where

$$B = \begin{pmatrix} 0.5(1 \pm \beta_1) & 0 & \cdots & 0 \\ 0 & 0.5(1 \pm \beta_2) & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0.5(1 \pm \beta_D) \end{pmatrix} . \tag{20}$$

Therefore,

$$h_{sbx}(\mathbf{x}_1 M, \mathbf{x}_2 M) = \mathbf{x}_1 M + (\mathbf{x}_2 - \mathbf{x}_1)MB \tag{21}$$

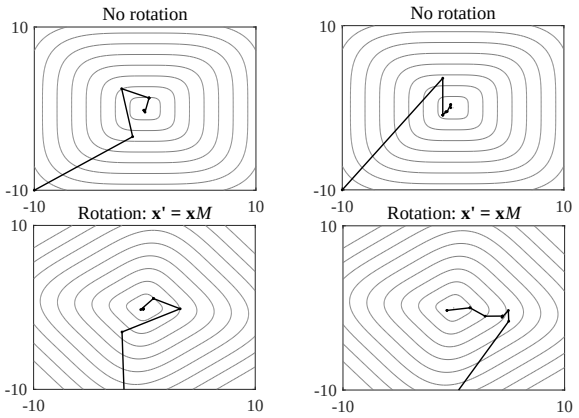
and

$$h_{sbx}(\mathbf{x}_1, \mathbf{x}_2)M = \mathbf{x}_1 M + (\mathbf{x}_2 - \mathbf{x}_1)BM \ . \tag{22}$$

That is, $h_{sbx}(\mathbf{x}_1 M, \mathbf{x}_2 M) = h_{sbx}(\mathbf{x}_1, \mathbf{x}_2)M$ holds only if $MB = BM$, i.e., $\beta_1 = \beta_2 = \ldots = \beta_D$, which is almost impossible since they are independent random numbers. Hence, the SBX operator is not rotation invariant.

Fig. 3 shows the convergence profiles of the mutation based DE and SBX based GA on $f(x_1, x_2) = x_1^4 + x_2^4$ with and without rotation. As can be seen, the search behavior of DE keeps unchanged on the two problems, where the convergence profile is rotated together with the search space. On the contrary, the search behavior of GA is unstable and the convergence profiles are different on the two problems. As can be further observed from Table 3, the performance of DE is similar to and much better than GA on six benchmark problems with and without rotation, respectively, which is consistent with the facts that the mutation operator of DE is rotation invariant while the SBX operator is not. In fact, the superiority of DE on problems with complicated variable linkages[18] is mainly

due to its rotation invariance property.



(a) DE (rotation invariant) (b) SBX based GA

Fig. 3: Convergence profiles of DE and SBX based GA with fixed random number seed on $f(x_1, x_2) = x_1^4 + x_2^4$ with and without rotation, where M is a randomly generated orthogonal matrix

Table 3: Minimum objective values obtained by DE and SBX based GA on six problems with and without rotation

Original problem ($x_1, \ldots, x_{30} \in [-10, 10]$)	DE (rotation invariant)	SBX based GA
Schwefel's Function 2.22	1.9799e+1 (2.08e+0)	5.4310e+0 (2.13e+0)
Schwefel's Function 2.21	1.5589e+0 (1.87e-1)	4.5792e+0 (1.21e+0)
Quaric Function	1.8303e+2 (9.67e+1)	1.6425e+3 (1.03e+3)
Griewank Function	4.9032e-1 (1.01e-1)	4.7700e-1 (1.87e-1)
Ackley's Function	3.7861e+0 (2.96e-1)	3.3728e+0 (3.75e-1)
Rastrigin's Function	2.4783e+2 (2.34e+1)	4.2269e+1 (1.08e+1)
Rotated problem ($\mathbf{x}' = \mathbf{x}M$)	DE (rotation invariant)	SBX based GA
Schwefel's Function 2.22	1.7401e+1 (3.38e+0)	2.8069e+1 (4.32e+0)
Schwefel's Function 2.21	1.5535e+0 (2.21e-1)	2.0725e+0 (4.82e-1)
Quaric Function	1.2403e+2 (7.80e+1)	9.8479e+2 (1.24e+3)
Griewank Function	4.6714e-1 (7.22e-2)	4.9478e-1 (1.20e-1)
Ackley's Function	3.8477e+0 (2.24e-1)	4.0805e+0 (5.49e-1)
Rastrigin's Function	2.4588e+2 (1.23e+1)	2.5095e+2 (1.12e+1)

3 Sufficient and Necessary Condition of Achieving Invariance Properties

It can be concluded from the above analysis that the three invariance properties are critical to the robustness of operators, and we can judge whether an operator possesses these properties according to their definitions. However, it is still tricky to consider them in the design of new operators explicitly. Therefore, this section deduces the generic form of translation, scale, and rotation invariant operators by three steps.

3.1 Theoretical Derivation

Firstly, let $\mathbf{w} = (w_1, w_2, \ldots) = (x_{1d}, x_{2d}, \ldots)$, then a translation invariant operator should satisfy

$$h(\mathbf{w} + b) = h(\mathbf{w}) + b \ . \tag{23}$$

To find the generic form of $h(\mathbf{w})$, it is first expanded in a first order Taylor series at an arbitrary point $\mathbf{w}_0$:

$$h(\mathbf{w}) = h(\mathbf{w}_0) + (\mathbf{w} - \mathbf{w}_0)\nabla h(\mathbf{w}_0') \ , \tag{24}$$

where $\nabla = (\frac{\partial}{\partial w_1}, \frac{\partial}{\partial w_2}, \ldots)^T$ and $\mathbf{w}_0'$ is an unknown point. Based on (23), we have

$$\begin{aligned} &h(\mathbf{w}_0) + (\mathbf{w} + b - \mathbf{w}_0)\nabla h(\mathbf{w}_0') \\ &\quad = h(\mathbf{w}_0) + (\mathbf{w} - \mathbf{w}_0)\nabla h(\mathbf{w}_0'') + b \ , \end{aligned} \tag{25}$$

where $\mathbf{w}_0'$ and $\mathbf{w}_0''$ are unknown points determined by $\mathbf{w}_0$. Since (25) holds for any b, we only consider the components including b in (25):

$$bI\nabla h(\mathbf{w}_0') = b \ , \tag{26}$$

where I is a vector of ones. That is,

$$I\nabla h(\mathbf{w}_0') = 1 \tag{27}$$

holds for any $\mathbf{w}_0'$, which measn that

$$\frac{\partial h}{\partial w_1} + \frac{\partial h}{\partial w_2} + \cdots = 1 \ . \tag{28}$$

Obviously, (28) is a quasilinear first-order nonhomogeneous partial differential equation[35]. Let $h = h(\mathbf{w})$ be the solution of (28) determined by a function $g(\mathbf{w}, h) = 0$, then the following homogeneous partial differential equation can be obtained:

$$\frac{\partial g}{\partial h} + \frac{\partial g}{\partial w_1} + \frac{\partial g}{\partial w_2} + \cdots = 0 \ , \tag{29}$$

and thus the following first integrals can be obtained:

$$\begin{cases} h - w_1 = c_1 \\ w_1 - w_2 = c_2 \\ w_2 - w_3 = c_3 \\ \cdots \quad \cdots \end{cases} , \tag{30}$$

Therefore, the solution of (29) is

$$g = g(h - w_1, w_1 - w_2, w_2 - w_3, \ldots) \ . \tag{31}$$

Since $g = 0$, there must exists a function ψ such that

$$h - w_1 = \psi(w_1 - w_2, w_2 - w_3, \ldots) \ , \tag{32}$$

which is equivalent to

$$h(\mathbf{w}) = w_1 + \psi(w_2 - w_1, w_3 - w_2, \ldots) \ . \tag{33}$$

Moreover, it is obvious that (33) satisfies (23), hence (33) is a sufficient and necessary condition of (23), and the following theorem can be given:

Theorem 1 (**Translation invariant operator**). *A continuously differentiable variation operator $h(x_{1d}, x_{2d}, \ldots)$ is translation invariant if and only if it has the following form:*

$$h(x_{1d}, x_{2d}, \ldots) = x_{1d} + \psi(x_{2d} - x_{1d}, x_{3d} - x_{2d}, \ldots) \ , \quad (34)$$

where ψ can be any continuously differentiable function.

Secondly, a scale invariant operator should satisfy

$$h(a\mathbf{w}) = a \cdot h(\mathbf{w}) \ , \quad (35)$$

according to (33), a translation and scale invariant operator should satisfy

$$\begin{aligned} &aw_1 + \psi(aw_2 - aw_1, aw_3 - aw_2, \ldots) \\ &\quad = aw_1 + a \cdot \psi(w_2 - w_1, w_3 - w_2, \ldots) \ . \end{aligned} \quad (36)$$

Let $\mathbf{v} = (v_1, v_2, \ldots) = (w_2 - w_1, w_3 - w_2, \ldots)$, we have

$$\psi(a\mathbf{v}) = a \cdot \psi(\mathbf{v}), \quad (37)$$

and the first order Taylor series expansion at an arbitrary point $\mathbf{v}_0$ is

$$\begin{aligned} &\psi(\mathbf{v}_0) + (a\mathbf{v} - \mathbf{v}_0)\nabla\psi(\mathbf{v}_0') \\ &\quad = a\psi(\mathbf{v}_0) + a(\mathbf{v} - \mathbf{v}_0)\nabla\psi(\mathbf{v}_0'') \ , \end{aligned} \quad (38)$$

where $\mathbf{v}_0'$ and $\mathbf{v}_0''$ are unknown points determined by $\mathbf{v}_0$. Since (38) holds for any a, we only consider the components excluding a in (38), that is,

$$\psi(\mathbf{v}_0) = \mathbf{v}_0\nabla\psi(\mathbf{v}_0') \quad (39)$$

must hold for any $\mathbf{v}_0$ and $\mathbf{v}_0'$, which means that

$$v_1\frac{\partial\psi}{\partial v_1} + v_2\frac{\partial\psi}{\partial v_2} + \cdots = \psi \ . \quad (40)$$

Let $g(\mathbf{v}, \psi) = 0$, the following homogeneous partial differential equation can be obtained:

$$\psi\frac{\partial g}{\partial\psi} + v_1\frac{\partial g}{\partial v_1} + v_2\frac{\partial g}{\partial v_2} + \cdots = 0 \ , \quad (41)$$

and thus the following first integrals can be obtained:

$$\begin{cases} \ln\psi - \ln v_1 = c_1 \\ \ln v_1 - \ln v_2 = c_2 \\ \ln v_2 - \ln v_3 = c_3 \\ \quad \cdots \quad \cdots \end{cases}, \quad (42)$$

Therefore, the solution of (41) is

$$g = g(\ln\psi - \ln v_1, \ln v_1 - \ln v_2, \ln v_2 - \ln v_3, \ldots) \ . \quad (43)$$

Since $g = 0$, there must exists a function φ such that

$$\ln\psi - \ln v_1 = \varphi(\ln v_1 - \ln v_2, \ln v_2 - \ln v_3, \ldots) \ , \quad (44)$$

hence the generic form of $\psi(\mathbf{v})$ can be determined:

$$\ln\psi(\mathbf{v}) = \ln v_1 + \varphi\left(\ln\frac{v_1}{v_2}, \ln\frac{v_2}{v_3}, \ldots\right) \ . \quad (45)$$

Let $\varphi(u_1, u_2, \ldots) = \ln\phi(e^{\frac{1}{u_1}}, e^{\frac{1}{u_2}}, \ldots)$, then

$$\psi(\mathbf{v}) = v_1\phi\left(\frac{v_2}{v_1}, \frac{v_3}{v_2}, \ldots\right) \ . \quad (46)$$

According to (33), we have

$$h(\mathbf{w}) = w_1 + (w_2 - w_1)\phi\left(\frac{w_3 - w_2}{w_2 - w_1}, \frac{w_4 - w_3}{w_3 - w_2}, \ldots\right). \quad (47)$$

Moreover, it is obvious that (47) satisfies both (23) and (35), hence (47) is a sufficient and necessary condition of (23) and (35), and the following theorem can be given:

Theorem 2 (**Translation and scale invariant operator**). *A continuously differentiable variation operator $h(x_{1d}, x_{2d}, \ldots)$ is translation and scale invariant if and only if it has the following form:*

$$h(x_{1d}, x_{2d}, \ldots) = x_{1d} + (x_{2d} - x_{1d})\phi\left(\frac{x_{3d} - x_{2d}}{x_{2d} - x_{1d}}, \frac{x_{4d} - x_{3d}}{x_{3d} - x_{2d}}, \ldots\right), \quad (48)$$

where ϕ can be any continuously differentiable function.

Thirdly, a rotation invariant operator should satisfy

$$h\left(\sum_{i=1}^{D} m_{id}x_{1i}, \sum_{i=1}^{D} m_{id}x_{2i}, \ldots\right) = \sum_{i=1}^{D} m_{id} \cdot h(x_{1i}, x_{2i}, \ldots) \ , \quad (49)$$

where $m_{id} \in M$ and $d = 1, \ldots, D$. Let $\phi(\mathbf{w}) = \varphi(w_1, \prod_{i=1}^{2} w_i, \prod_{i=1}^{3} w_i, \ldots)$, then (47) is equivalent to

$$h(\mathbf{w}) = w_1 + (w_2 - w_1)\varphi\left(\frac{w_3 - w_2}{w_2 - w_1}, \frac{w_4 - w_3}{w_2 - w_1}, \ldots\right) \ . \quad (50)$$

According to (49), a translation, scale, and rotation invariant operator should satisfy

$$\begin{aligned} &\sum_{i=1}^{D} m_{id}x_{1i} + \left(\sum_{i=1}^{D} m_{id}x_{2i} - \sum_{i=1}^{D} m_{id}x_{1i}\right) \\ &\cdot \varphi\left(\frac{\sum_{i=1}^{D} m_{id}x_{3i} - \sum_{i=1}^{D} m_{id}x_{2i}}{\sum_{i=1}^{D} m_{id}x_{2i} - \sum_{i=1}^{D} m_{id}x_{1i}}, \ldots\right) \\ &= \sum_{i=1}^{D} m_{id}\left[x_{1i} + (x_{2i} - x_{1i})\varphi\left(\frac{x_{3i} - x_{2i}}{x_{2i} - x_{1i}}, \ldots\right)\right] \ , \end{aligned} \quad (51)$$

which is equivalent to

$$\begin{aligned} &\sum_{i=1}^{D} m_{id}(x_{2i} - x_{1i}) \cdot \varphi\left(\frac{\sum_{i=1}^{D} m_{id}(x_{3i} - x_{2i})}{\sum_{i=1}^{D} m_{id}(x_{2i} - x_{1i})}, \ldots\right) \\ &= \sum_{i=1}^{D} m_{id}(x_{2i} - x_{1i})\varphi\left(\frac{m_{id}(x_{3i} - x_{2i})}{m_{id}(x_{2i} - x_{1i})}, \ldots\right) \ . \end{aligned} \quad (52)$$

Let $\mathbf{u}_i = (u_{i1}, u_{i2}, \ldots) = (m_{id}(x_{2i} - x_{1i}), m_{id}(x_{3i} - x_{2i}), \ldots)$, we have

$$\sum_{i=1}^{D} u_{i1} \cdot \varphi\left(\frac{\sum_{i=1}^{D} u_{i2}}{\sum_{i=1}^{D} u_{i1}}, \ldots\right) = \sum_{i=1}^{D} u_{i1}\varphi\left(\frac{u_{i2}}{u_{i1}}, \ldots\right) \ , \quad (53)$$

and the first order Taylor series expansion at an arbitrary point $\mathbf{u}_0 = (u_{01}, u_{02}, \ldots)$ is

$$\begin{aligned}&\sum_{i=1}^{D} u_{i1} \cdot \left[\varphi(\mathbf{u}_0) + \left(\frac{\sum_{i=1}^{D} u_{i2}}{\sum_{i=1}^{D} u_{i1}} - u_{01}, \ldots\right) \nabla\varphi(\mathbf{u}_0')\right] \\ &= u_{11} \cdot \left[\varphi(\mathbf{u}_0) + \left(\frac{u_{12}}{u_{11}} - u_{01}, \ldots\right) \nabla\varphi(\mathbf{u}_0'')\right] \\ &+ u_{21} \cdot \left[\varphi(\mathbf{u}_0) + \left(\frac{u_{22}}{u_{21}} - u_{01}, \ldots\right) \nabla\varphi(\mathbf{u}_0''')\right] \\ &+ \cdots ,\end{aligned} \tag{54}$$

which can be simplified as

$$\begin{aligned}&\left(\sum_{i=1}^{D} u_{i2} - u_{01} \sum_{i=1}^{D} u_{i1}, \ldots\right) \nabla\varphi(\mathbf{u}_0') \\ &\quad = (u_{12} - u_{01}u_{11}, \ldots) \nabla\varphi(\mathbf{u}_0'') \\ &\quad + (u_{22} - u_{01}u_{21}, \ldots) \nabla\varphi(\mathbf{u}_0''') \\ &\quad + \cdots ,\end{aligned} \tag{55}$$

where $\mathbf{u}_0', \mathbf{u}_0'', \ldots$ are unknown points determined by $\mathbf{u}_0$. Since (55) holds for any $u_{12}, u_{22}, \ldots$, we have

$$\nabla\varphi(\mathbf{u}_0') = \nabla\varphi(\mathbf{u}_0'') = \nabla\varphi(\mathbf{u}_0''') = \cdots = \mathbf{c} , \tag{56}$$

and the form of $\varphi(\mathbf{u})$ can only be

$$\varphi(\mathbf{u}) = c_0 + c_1 u_1 + c_2 u_2 + \cdots , \tag{57}$$

where $c_0, c_1, c_2, \ldots$ are constants. According to (50),

$$h(\mathbf{w}) = w_1 + c_0(w_2 - w_1) + c_1(w_3 - w_2) + c_2(w_4 - w_3) + \cdots . \tag{58}$$

Let

$$\begin{cases} 1 - c_0 = r_1 \\ c_0 - c_1 = r_2 \\ c_1 - c_2 = r_3 \\ \cdots \quad \cdots \end{cases} , \tag{59}$$

we have

$$h(\mathbf{w}) = r_1 w_1 + r_2 w_2 + r_3 w_3 + \cdots \tag{60}$$

and $r_1 + r_2 + r_3 + \cdots = 1$. Moreover, it is obvious that (60) satisfies (23), (35), and (49), hence (60) is a sufficient and necessary condition of (23), (35), and (49), and the following theorem can be given:

Theorem 3 (**Translation, scale, and rotation invariant operator**). *A continuously differentiable variation operator $h(x_{1d}, x_{2d}, \ldots)$ is translation, scale, and rotation invariant if and only if it has the following form:*

$$h(x_{1d}, x_{2d}, \ldots) = r_1 x_{1d} + r_2 x_{2d} + r_3 x_{3d} + \cdots , \tag{61}$$

where $r_1, r_2, r_3, \ldots$ can be any real constants satisfying $r_1 + r_2 + r_3 + \cdots = 1$.

3.2 Remarks

According to the above theorem, the following three corollaries can be given:

1. An operator satisfying (61) is scale invariant.
2. An operator satisfying (61) with $r_1 + r_2 + r_3 + \cdots = 1$ is scale and translation invariant.
3. An operator satisfying (61) with $r_1, r_2, r_3, \ldots$ being constants is scale and rotation invariant.

The theoretical proofs of these corollaries are given in the following.

Corollary 1. *If a variation operator satisfying*

$$h(x_{1d}, x_{2d}, \ldots) = r_1 x_{1d} + r_2 x_{2d} + r_3 x_{3d} + \cdots , \tag{62}$$

then it is scale invariant.

Proof. Since

$$\begin{aligned}h(ax_{1d}, ax_{2d}, \ldots) &= ar_1 x_{1d} + ar_2 x_{2d} + ar_3 x_{3d} + \cdots \\ &= a h(x_{1d}, x_{2d}, \ldots) ,\end{aligned} \tag{63}$$

the operator is scale invariant. □

Corollary 2. *If a variation operator satisfying*

$$h(x_{1d}, x_{2d}, \ldots) = r_1 x_{1d} + r_2 x_{2d} + r_3 x_{3d} + \cdots \tag{64}$$

with $r_1 + r_2 + r_3 + \cdots = 1$, then it is scale and translation invariant.

Proof. Since

$$\begin{aligned}&h(x_{1d} + b, x_{2d} + b, \ldots) \\ &\quad = r_1(x_{1d} + b) + r_2(x_{2d} + b) + r_3(x_{3d} + b) + \cdots \\ &\quad = r_1 x_{1d} + r_2 x_{2d} + r_3 x_{3d} + \cdots + (r_1 + r_2 + r_3 + \cdots)b \\ &\quad = h(x_{1d}, x_{2d}, \ldots) + b ,\end{aligned} \tag{65}$$

according to Corollary 1, the operator is scale and translation invariant. □

Corollary 3. *If a variation operator satisfying*

$$h(x_{1d}, x_{2d}, \ldots) = r_1 x_{1d} + r_2 x_{2d} + r_3 x_{3d} + \cdots \tag{66}$$

with $r_1, r_2, r_3, \ldots$ being constants, then it is scale and rotation invariant.

Proof. Since $r_1, r_2, r_3, \ldots$ are constants, we have

$$h(\mathbf{x}_1, \mathbf{x}_2, \ldots) = r_1 \mathbf{x}_1 + r_2 \mathbf{x}_2 + r_3 \mathbf{x}_3 + \cdots . \tag{67}$$

Therefore,

$$\begin{aligned}&h(\mathbf{x}_1 M, \mathbf{x}_2 M, \ldots) \\ &\quad = r_1(\mathbf{x}_1 M) + r_2(\mathbf{x}_2 M) + r_3(\mathbf{x}_3 M) + \cdots \\ &\quad = (r_1 \mathbf{x}_1)M + (r_2 \mathbf{x}_2)M + (r_3 \mathbf{x}_3)M + \cdots \\ &\quad = h(\mathbf{x}_1, \mathbf{x}_2, \ldots)M ,\end{aligned} \tag{68}$$

according to Corollary 1, the operator is scale and rotation invariant. □

Generally, most existing operators including those in GA, DE, and CMA-ES meet the second corollary and are scale and translation invariant. However, the operators in GA and CMA-ES are not rotation invariant since the weights (i.e., β in (2) and $\mathcal{N}(0, {\sigma'_d}^2 c)$ in (12)) vary on different dimensions. By contrast, the mutation operator of DE is rotation invariant since the weights (i.e., F in (3)) keep unchanged on all dimensions. Nevertheless, it should be noted that a rotation invariant operator does not necessarily lead to a rotation invariant metaheuristic, and vice versa. For example, the mutation operator of DE is rotation variant, but DE is not rotation invariant due to the crossover operator with $CR < 1$[22]. The operator of CMA-ES is not rotation invariant, but CMA-ES is approximately rotation invariant due to the rotation angle adaptive covariance matrix C[8]. In fact, a rotation invariant operator may not obtain good performance since offspring solutions can only be the linear combinations of parents.

For the sake of robustness and generalization, it is necessary to consider translation invariance, scale invariance, and rotation invariance in designing new operators, which can exhibit the same performance on the same landscape with arbitrary search spaces. Nevertheless, many existing operators meet Corollary 2 but are not strictly rotation invariant (i.e., the weights $r_1, r_2, r_3, \ldots$ are not constant), aiming to balance between rotation invariance and search performance. In practice, rotation invariance is approximately achieved by rotating offspring solutions according to a covariance matrix [8, 36] or associating each weight with multiple candidate constants [7].

4 Principled Approach for Designing Operators

The function of (61) reveals the generic form of translation, scale, and rotation invariant operators, while the optimal values of the weights $r_1, r_2, r_3, \ldots$ are not provided. Therefore, this section proposes a principled approach for designing new operators, which searches for high-performance operators by optimizing the weights.

4.1 Parameterization of Variation Operators

In order to introduce randomness, the proposed approach AutoV represents each weight by an independent normal distribution, and it optimizes the mean and variance of each normal distribution instead of the weight. Formally, the proposed AutoV searches for the operators having the following form:

$$\begin{aligned} h(x_{1d}, \ldots, x_{td}) &= \sum_{i=1}^{t} r_i x_{id} \\ \text{s.t.} \quad \sum_{i=1}^{t} r_i &= 1, \ r_i \sim \mathcal{N}(\mu_i, \sigma_i^2) \end{aligned}, \tag{69}$$

where $\mu_i \in [-1, 1]$ and $\sigma_i \in [0, 1]$ are the parameters to be optimized. Note that the value of r_i keeps unchanged for $d = 1, \ldots, D$, hence r_i is still a constant for generating each offspring solution.

In order to balance between rotation invariance and search performance, the ensemble of multiple parameter sets is adopted in AutoV. Moreover, the first weight r_1 can be omitted since it can be directly set to $1 - r_2 - r_3 - \cdots$. To summarize, an operator in AutoV is determined by the following matrix

$$\begin{pmatrix} \mu_{12}, \sigma_{12}, \mu_{13}, \sigma_{13}, \ldots, \mu_{1t}, \sigma_{1t}, p_1 \\ \mu_{22}, \sigma_{22}, \mu_{23}, \sigma_{23}, \ldots, \mu_{2t}, \sigma_{2t}, p_2 \\ \cdots \quad \cdots \\ \mu_{k2}, \sigma_{k2}, \mu_{k3}, \sigma_{k3}, \ldots, \mu_{kt}, \sigma_{kt}, p_k \end{pmatrix}, \tag{70}$$

where μ_{ji} and σ_{ji} denote the mean and variance of weight r_i in the j-th parameter set, respectively, and p_j denotes the probability of selecting the j-th parameter set. When generating a decision variable of an offspring solution, the roulette-wheel selection is first used to select a parameter set (i.e., one row of (70)) according to their probabilities. Then, the decision variable is generated by sampling the given normal distribution.

In this way, the search of high-performance operators can be formulated as a continuous optimization problem, whose decision vector contains all the elements in (70) and the objective is the performance of the corresponding operator. Generally, it is easy to measure the fitness by investigating the performance of a metaheuristic equipped with the operator, but it is difficult to optimize the decision vector since AutoV does not require any prior knowledge, i.e., none of the existing optimizers are used to evolve a population for solving the problem. Thus, the proposed AutoV suggests a novel evolutionary procedure, in which the population can be evolved by itself.

4.2 Procedure of the Principled Approach

The procedure of the proposed AutoV is detailed in Fig. 4 and Algorithm 1. Given a benchmark problem for performance measurement, AutoV first randomly initializes a population P and evaluates each solution in P, where each solution denotes a parameter matrix defined in (70). At each generation, AutoV selects a number of parents from P via binary tournament selection, then uses the

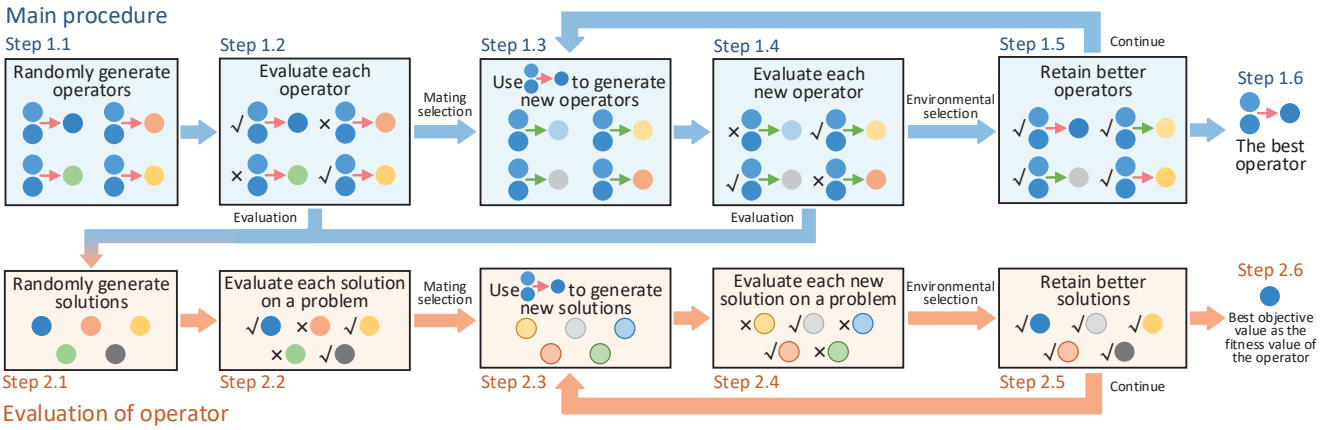


Fig. 4: Procedure of AutoV

Algorithm 1: Procedure of AutoV

Input: f (a benchmark problem)
Output: $\mathbf{h}$ (the best operator)

$P \leftarrow$ Randomly initialize a population, where each solution is a parameter matrix defined in (70);
for *each* $\mathbf{x} \in P$ **do**
  Evaluate the objective value of $\mathbf{x}$ by $Evaluation(\mathbf{x}, f)$;
while *termination criterion is not fulfilled* **do**
  $P' \leftarrow$ Select parents from P via binary tournament selection;
  $\mathbf{h} \leftarrow$ The best solution in P;
  $O \leftarrow$ Use the operator $\mathbf{h}$ to generate offspring based on parents P';
  for *each* $\mathbf{o} \in O$ **do**
    Evaluate the objective value of $\mathbf{o}$ by $Evaluation(\mathbf{o}, f)$;
  $P \leftarrow P \cup O$;
  $P \leftarrow$ Half the solutions in P with better fitness;
$\mathbf{h} \leftarrow$ The best solution (i.e., operator) in P;
return $\mathbf{h}$;

parents to generate an offspring population by using the variation operator parameterized by the best solution in P. Afterwards, the offspring population is combined with P, and half the solutions with better fitness survive for the next generation. AutoV repeats the above steps until the termination criterion is fulfilled, and returns the solution with the best fitness as the found variation operator.

Since the goal of AutoV is to create high-performance variation operators, it uses the best operator found so far to evolve the population for finding a better operator, and the better operator can then evolve the population for finding a much better operator, where none of the existing metaheuristics are used. For the fitness evaluation of each candidate operator, a simple metaheuristic is established by adopting the candidate operator. As presented in Algorithm 2, the best objective value found by the established metaheuristic on the given benchmark problem is regarded as the fitness of the candidate operator. Besides, we find that a candidate operator may be over-optimized for the given benchmark problem, which means that it considers a perfect solution for the given benchmark problem accidentally. Still, it cannot evolve the population gradually. To solve this issue, AutoV executes the established metaheuristic for multiple runs and uses the median value of the found best objective values as the fitness.

In the experiments, we consider the following five different sets of parents as the input of (69) for generating one offspring solution:

$$\begin{cases} h_1 = h(x_{1d}, x_{2d}) \\ h_2 = h(x_{1d}, l_d, u_d) \\ h_3 = h(x_{1d}, x_{2d}, l_d, u_d) \\ h_4 = h(x_{1d}, x_{2d}, x_{3d}) \\ h_5 = h(x_{1d}, x_{2d}, x_{3d}, l_d, u_d) \end{cases}, \tag{71}$$

Algorithm 2: *Evaluation*$(\mathbf{p}, f)$

Input: $\mathbf{p}$ (parameter matrix of an operator), f (a benchmark problem)
Output: fit (fitness of operator $\mathbf{p}$)

$fit \leftarrow \emptyset$;
for $run = 1$ *to* $maxRun$ **do**
  $P \leftarrow$ Randomly initialize a population, where each solution is a decision vector for f;
  for *each* $\mathbf{x} \in P$ **do**
    Evaluate the objective of $\mathbf{x}$ by $f(\mathbf{x})$;
  while *termination criterion is not fulfilled* **do**
    $P' \leftarrow$ Select parents from P via binary tournament selection;
    $O \leftarrow$ Use the operator $\mathbf{p}$ to generate offspring based on parents P';
    for *each* $\mathbf{o} \in O$ **do**
      Evaluate the objective of $\mathbf{o}$ by $f(\mathbf{o})$;
    $P \leftarrow P \cup O$;
    $P \leftarrow$ Half the solutions in P with better fitness;
  $\mathbf{x} \leftarrow$ The best solution in P;
  $fit \leftarrow fit \cup \{f(\mathbf{x})\}$;
$fit \leftarrow$ Median value of fit;
return fit;

where x_{1d}, x_{2d}, x_{3d} denote the decision variables of three parents, l_d denotes the lower bound, and u_d denotes the upper bound on the d-th dimension.

It can be found from Algorithm 2 that a metaheuristic with an operator found by AutoV holds a similar procedure to general genetic algorithms, where the time complexity of mating selection is $O(N)$ and the time complexity of environmental selection is $O(N \log N)$ (N is the population size). By contrast, the offspring generation becomes less efficient due to the use of multiple parameter sets, whose time complexity is $O(NDk)$ (D is the number of decision variables and k is the number of parameter sets). Therefore, the time complexity of one generation of a metaheuristic with an operator found by AutoV is $O(NDk)$. On the other hand, the time complexity of one generation of AutoV itself is as high as $O(N^2DkG \times maxRun)$, since N metaheuristics need to be executed for $maxRun$ times and each time contains G generations. Nevertheless, the search of high-performance operators in AutoV is an off-line procedure that does not affect the efficiency of solving problems in practice.

5 Experimental Studies

To verify the effectiveness of the proposed AutoV, the performance of the operators found by AutoV is first studied. Then, the best operator is compared with eight metaheuristics, where GA[6], PSO[10], DE[7], CMA-ES[8], and FEP[9] are classical metaheuristics, CSO[37] is a competitive swarm optimizer for large-scale optimization, and SHADE[38] and IMODE[39] are hybrid metaheuristics based on multiple operators with parameter adaptation. Based on the settings suggested in the original literature of the compared metaheuristics, we finely tune their parameters for a relatively good performance, where the detailed parameter settings are listed in Table 4.

Table 4: Parameter settings of the compared metaheuristics, where D is the number of decision variables, $\mathbf{u}$ is the upper bound, $\mathbf{l}$ is the lower bound, and $n = 100$ is the population size

Metaheuristic	Parameter setting
GA[6] (based on SBX[32] and polynomial mutation[40])	crossover probability: 1, mutation probability: $1/D$, distribution index: 20
PSO[10]	inertia weight: 0.4
DE[7] (DE/rand/1/bin)	$CR = 0.9$, $F = 0.5$
CMA-ES[8]	initial $\sigma = 0.6(\mathbf{u} - \mathbf{l})$, $w_i = \log(\mu + 0.5) - \log(i)$, $i \in [1, \mu]$, $\mu = 0.5n$, $c_\sigma = 0.15$, $d_\sigma = 1.15$, $c_c = 0.105$
FEP[9]	initial $\eta = 3$
CSO[37]	social factor: 0.1
SHADE[38] (parameter adaptative DE)	–
IMODE[39] (winner of CEC'2020)	Minimum population size: 4 Ratio of archive size: 2.6

5.1 Comparison Between the Operators Found by AutoV

For each of the five functions given in (71), the proposed AutoV optimizes it with a population size of 100 for 1000

generations. As for the fitness evaluation of each candidate operator, the population size is set to 100, the number of generations is set to 100, the number of parameter sets k is set to 10, and the number of runs $maxRun$ is set to 9. Besides, the Rastrigin's Function[17] is adopted as the benchmark problem for performance measurement.

To compare the performance of the found operators with different functions h_1–h_5, they are embedded in the simple metaheuristic presented in Algorithm 2 and tested on eight benchmark problems with 30 decision variables. These benchmark problems have a variety of unimodal, multimodal, or flat landscapes, whose definitions can be found in Ref.[17]. For all the metaheuristics, the population size is set to 100 and the number of generations is set to 100. Table 5 lists the minimum objective values found by the five metaheuristics averaged over 30 runs, where the compared operators exhibit similar performance and the operator h_3 has slightly better overall performance than the others. It is worth noting that the operator h_5 does not obtain the best overall performance, though the function h_5 has more parents and is expected to perform better. This is because the function h_5 contains more parameters to be optimized, which hinders AutoV from finding high-performance operators.

Table 5: Minimum objective values obtained by the operators with different functions h_1–h_5 found by AutoV on eight benchmark problems. Best results are highlighted

Problem	h_1	h_2	h_3	h_4	h_5
Schwefel's Function 2.22	9.07e-5	5.27e+0	2.07e-2	8.72e-3	1.02e-1
Schwefel's Function 2.21	1.65e+1	1.19e+1	1.66e+0	8.75e+0	1.67e+0
Quartic Function	2.01e-2	1.32e-1	1.12e-2	9.02e-3	1.16e-2
Generalized Griewank Function	1.99e-3	2.93e+0	2.81e-2	4.46e-1	2.66e-1
Generalized Schwefel's Function 2.26	-9.17e+3	-1.04e+4	-1.16e+4	-5.70e+3	-5.28e+3
Ackley's Function	5.47e-4	4.77e+0	1.79e-2	8.94e-1	2.05e-1
Rosenbrock's Function	1.20e+4	1.17e+5	9.09e+3	8.80e+3	3.82e+3
Rastrigin's Function	3.76e+1	3.96e+1	3.88e+0	9.99e+0	7.86e+0

To study the influence of the number of parameter sets k, Fig. 5 plots the performance of the metaheuristic with operator h_3 and $k = 1, 5, 10, 15, 20$ on the eight benchmark problems, where $k = 10$ leads to better overall performance than the other settings. On the one hand, a small value of k provides a few different search behaviors, and thus leads to a low performance limit. On the other hand, although a large value of k provides many different search behaviors, it leads to a large number of parameters that are difficult to be optimized. As a consequence, $k = 10$ is a proper setting for finding high-performance operators.

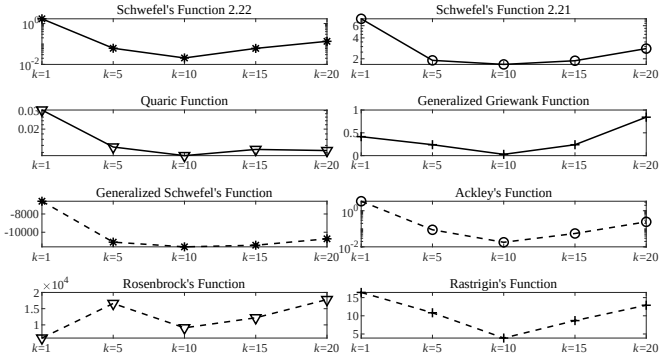


Fig. 5: Minimum objective values obtained by the operator with function h_3 and different numbers of parameter sets k found by AutoV

Furthermore, the influence of the benchmark problem on fitness evaluation is studied. Table 6 lists the performance of the metaheuristic with operator h_3 optimized on four benchmark problems, including the Schwefel's Function 2.22 with a unimodal landscape, the Schwefel's Function 2.21 with a flat landscape, and the Ackley's Function and Rastrigin's Function with multimodal landscapes. It can be found that the four metaheuristics obtain the best performance on the benchmark problem for fitness evaluation, while they obtain quite similar performance on the other problems. In the following, the metaheuristic with operator h_3 and $k = 10$ optimized on the Rastrigin's Function is used as a representative of AutoV to be compared with existing metaheuristics on various problems.

Table 6: Minimum objective values obtained by the operator with function h_3 and different benchmark problems for fitness evaluation. Best results are highlighted

Problem	Optimized on Schwefel's Function 2.22	Optimized on Schwefel's Function 2.21	Optimized on Ackley's Function	Optimized on Rastrigin's Function
Schwefel's Function 2.22	2.46e-6	7.00e-2	4.91e-6	2.07e-2
Schwefel's Function 2.21	1.46e+1	1.36e+0	1.21e+1	1.66e+0
Quartic Function	1.29e-2	1.23e-2	1.44e-2	1.12e-2
Generalized Griewank Function	6.89e-3	7.02e-2	1.48e-3	2.81e-2
Generalized Schwefel's Function 2.26	-9.35e+3	-9.76e+3	-9.55e+3	-1.16e+4
Ackley's Function	1.00e-4	5.77e-2	5.71e-5	1.79e-2
Rosenbrock's Function	6.03e+3	3.89e+3	7.64e+3	9.09e+3
Rastrigin's Function	2.36e+1	1.59e+1	2.91e+1	3.88e+0

5.2 Comparison on Small-Scale Benchmark Problems

Then, the proposed AutoV (i.e., the metaheuristic with operator h_3) is compared with eight existing metaheuristics on 13 small-scale benchmark problems with 30 decision variables, where the definitions of these problems

Table 7: Minimum objective values obtained by nine metaheuristics on 13 small-scale benchmark problems with 30 decision variables. Best results are highlighted

Small-scale problems[9]	GA[6]	PSO[10]	DE[7]	CMA-ES[8]	FEP[9]	CSO[37]	SHADE[38]	IMODE[39]	AutoV
f_1	1.4973e+1 − (2.0053e+0)	3.5453e+3 − (1.2242e+3)	5.0032e+3 − (5.7205e+2)	1.2391e+2 − (2.6004e+1)	9.2809e+3 − (2.7503e+3)	1.2988e+3 − (5.0772e+2)	2.2958e+1 − (7.4054e+0)	2.9105e+0 − (9.0907e-1)	8.3702e-1 (5.333e-1)
f_2	8.3310e-1 − (1.7114e-1)	3.2170e+1 − (6.6473e+0)	5.1595e+1 − (1.5480e+1)	8.6103e+0 − (3.3770e+0)	5.1291e+1 − (1.0920e+1)	1.5504e+1 − (3.1662e+0)	6.2312e+0 − (1.1158e+0)	3.2305e-1 − (1.0412e-1)	1.2081e-1 (2.556e-2)
f_3	1.0621e+4 − (2.6732e+3)	7.8076e+3 − (2.9985e+3)	6.4156e+3 − (1.1614e+3)	5.0816e+3 − (1.2254e+3)	2.1959e+4 − (3.5707e+3)	1.7542e+3 ≈ (5.5302e+2)	1.6461e+3 ≈ (5.3479e+2)	6.6967e+2 + (1.3529e+2)	2.0131e+3 (8.307e+2)
f_4	2.3222e+1 − (3.6870e+0)	2.4005e+1 − (3.0366e+0)	3.4503e+1 − (4.1625e+0)	8.7765e+0 − (1.5806e+0)	5.3464e+1 − (9.3989e+0)	1.1521e+1 − (2.0034e+0)	9.1224e+0 − (1.4948e+0)	8.8346e+0 − (2.0574e+0)	6.4179e+0 (1.953e+0)
f_5	1.2336e+3 − (6.8051e+2)	5.5668e+5 − (2.7471e+5)	1.6919e+6 − (1.0907e+6)	5.1996e+3 − (2.8736e+3)	5.8033e+6 − (2.9157e+6)	1.5258e+5 − (7.7532e+4)	7.2347e+2 − (3.3376e+2)	1.2120e+2 ≈ (6.9918e+1)	2.0580e+2 (1.981e+2)
f_6	1.9375e+1 − (3.9978e+0)	3.4820e+3 − (1.1301e+3)	4.5314e+3 − (8.0852e+2)	1.3163e+2 − (3.2824e+1)	8.8923e+3 − (3.1734e+3)	1.0266e+3 − (4.3859e+2)	3.2250e+1 − (7.7965e+0)	3.6250e+0 − (2.1998e+0)	0.0000e+0 (0.000e+0)
f_7	9.1447e-2 − (3.8031e-2)	8.2203e-1 − (2.2065e-1)	1.1222e+0 − (3.6673e-1)	8.2239e-2 − (2.6534e-2)	5.2040e+1 − (2.7550e+1)	5.4577e-2 − (2.1989e-2)	5.8113e-2 − (1.4618e-2)	8.4790e-2 − (2.7932e-2)	2.0943e-2 (8.192e-3)
f_8	-1.4025e+4 ≈ (5.0347e+2)	-8.2308e+3 − (1.2979e+3)	-1.1684e+4 − (5.6921e+2)	-1.2799e+4 − (2.8077e+2)	-1.0531e+4 − (8.2223e+2)	-1.2285e+4 − (4.2013e+2)	-1.4023e+4 ≈ (8.5233e+1)	-1.4575e+4 + (2.0746e+2)	-1.3852e+4 (6.717e+2)
f_9	1.1504e+1 + (3.3186e+0)	1.5864e+2 − (1.7533e+1)	2.7333e+2 − (1.4085e+1)	2.2788e+2 − (2.3562e+1)	2.6914e+2 − (2.7653e+1)	1.0901e+2 − (1.7125e+1)	1.6771e+2 − (1.4089e+1)	2.3842e+1 − (5.6385e+0)	1.4546e+1 (2.163e+0)
f_{10}	1.7023e+0 − (3.1651e-1)	1.2329e+1 − (1.3322e+0)	1.3329e+1 − (8.5004e-1)	4.0097e+0 − (5.6144e-1)	1.4671e+1 − (7.3884e-1)	7.6077e+0 − (1.1400e+0)	2.8370e+0 − (2.1827e-1)	1.2297e+0 − (3.8807e-1)	2.1621e-1 (5.119e-2)
f_{11}	1.1563e+0 − (4.0584e-2)	2.8566e+1 − (3.2736e+0)	4.6051e+1 − (1.3901e+1)	2.1092e+0 − (2.8414e-1)	1.1710e+2 − (3.2432e+1)	1.0340e+1 − (3.2046e+0)	1.2644e+0 − (7.1606e-2)	1.0158e+0 − (3.6145e-2)	6.7928e-1 (1.999e-1)
f_{12}	2.8786e+1 − (1.5748e+1)	1.0624e+3 − (1.3982e+3)	7.1756e+4 − (6.5424e+4)	4.0254e+1 − (5.5667e+0)	4.9839e+6 − (5.9492e+6)	1.3627e+2 − (3.5766e+1)	3.1790e+1 − (9.1497e+0)	7.2029e+1 − (2.4378e+1)	1.5847e+0 (1.976e+0)
f_{13}	7.1895e+0 − (7.2480e+0)	4.3602e+5 − (5.1304e+5)	2.1898e+6 − (1.7255e+6)	1.0939e+1 − (2.7540e+0)	1.2093e+7 − (1.0145e+7)	2.1808e+3 − (5.2706e+3)	4.3094e+0 − (1.5238e+0)	4.3502e+0 − (2.3093e+0)	1.0075e-1 (4.216e-2)
+/−/≈	1/11/1	0/13/0	0/13/0	0/13/0	0/13/0	0/12/1	0/11/2	2/10/1	

'+', '−' and '≈' indicate that the result is significantly better, significantly worse, and statistically similar to that obtained by AutoV.

can be found in Ref.[9]. For all the compared metaheuristics, the population size is set to 100 and the number of function evaluations is set to 10000. Table 7 shows the means and standard deviations of the minimum objective values found by the nine metaheuristics, averaged over 30 runs. It can be found that the proposed AutoV obtains the best overall performance, which gains the best results on 9 out of 13 problems. The Wilcoxon rank sum test[41] with a significance level of 0.05 is adopted to perform statistical analysis, where AutoV is significantly better than GA, PSO, DE, CMA-ES, FEP, CSO, SHADE, IMODE on 11, 13, 13, 13, 13, 12, 11, and 10 problems, respectively. Furthermore, Fig. 6 depicts the convergence trajectories of the compared metaheuristics on the Step Function and the Penalized Function, where AutoV converges obviously faster than the other metaheuristics.

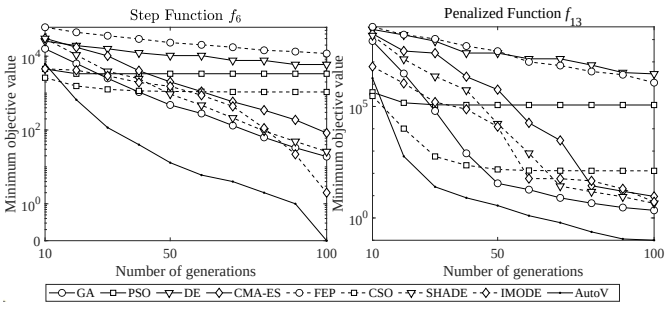


Fig. 6: Convergence trajectories of nine metaheuristics on two problems

While the 13 benchmark problems are already translated and scaled, they are further rotated by randomly generated orthogonal matrices and challenge the nine metaheuristics. As can be seen from the experimental results listed in Table 8, the superiority of the proposed AutoV becomes more significant, where AutoV outperforms the other metaheuristics on 11 out of 13 problems. As a consequence, the superiority of AutoV over some classical and state-of-the-art metaheuristics can be verified. Besides, it also implies that AutoV is more effective than the approaches based on the recommendation and combination of existing metaheuristics, whose performance can hardly go beyond the best existing metaheuristic on each problem.

5.3 Comparison on Large-Scale Benchmark Problems

Lastly, the proposed AutoV and the eight existing metaheuristics are compared on the 15 CEC'2013 large-scale benchmark problems[42]. These benchmark problems contain approximately 1000 decision variables and a variety of landscape functions, transformations, and interactions between variables, posing stiff challenges to general metaheuristics. For all the compared algorithms, the population size is set to 100 and the number of function evaluations is set to 120000.

Table 9 presents the means and standard deviations of the minimum objective values found by the compared

Table 8: Minimum objective values obtained by nine metaheuristics on 13 rotated small-scale benchmark problems with 30 decision variables. Best results are highlighted

Rotated small-scale problems[9]	GA[6]	PSO[10]	DE[7]	CMA-ES[8]	FEP[9]	CSO[37]	SHADE[38]	IMODE[39]	AutoV
f_1	1.7644e+1 − (5.3457e+0)	3.1210e+3 − (8.2974e+2)	4.3448e+3 − (8.3965e+2)	1.2422e+2 − (2.9570e+1)	9.2516e+3 − (3.9378e+3)	1.2894e+3 − (6.7999e+2)	2.9700e+1 − (1.0900e+1)	2.0650e+0 − (1.0838e+0)	7.1625e-1 (5.403e-1)
f_2	1.9315e+1 − (1.0234e+1)	3.7244e+1 − (1.8184e+1)	4.9671e+1 − (8.5749e+0)	1.3854e+1 − (6.8676e+0)	8.1606e+1 − (9.7066e+0)	1.4780e+1 − (3.0538e+0)	1.8746e+1 − (2.6580e+0)	1.5749e+0 ≈ (4.2566e-1)	1.5246e+0 (5.190e-1)
f_3	9.9581e+3 − (2.6366e+3)	5.3590e+3 − (1.4792e+3)	5.3369e+3 − (1.3071e+3)	1.3929e+4 − (3.9351e+3)	2.2100e+4 − (5.0673e+3)	1.5239e+3 ≈ (9.7521e+2)	1.7530e+3 ≈ (4.7904e+2)	8.5084e+2 + (2.5578e+2)	2.1193e+3 (7.960e+2)
f_4	1.4962e+1 − (5.9860e+0)	2.4756e+1 − (4.8244e+0)	3.6628e+1 − (4.7317e+0)	7.8044e+0 − (9.4482e-1)	4.6193e+1 − (2.4146e+0)	1.2863e+1 − (1.8236e+0)	8.1105e+0 − (8.4983e-1)	5.5831e+0 − (1.2362e+0)	3.2227e+0 (1.742e+0)
f_5	2.8300e+4 − (4.8151e+4)	8.9721e+5 − (3.4766e+5)	1.7625e+6 − (7.4240e+5)	9.3046e+3 − (6.0657e+3)	5.2830e+6 − (2.3632e+6)	1.1350e+5 − (6.7006e+4)	2.8513e+3 − (3.4473e+3)	2.9329e+2 ≈ (1.6130e+2)	2.0175e+2 (1.540e+2)
f_6	2.3375e+1 − (3.7009e+0)	3.6026e+3 − (6.8520e+2)	5.0828e+3 − (1.1280e+3)	1.1925e+2 − (2.9011e+1)	8.6534e+3 − (2.1819e+3)	1.2864e+3 − (5.8652e+2)	3.2250e+1 − (6.5629e+0)	1.3750e+1 − (7.1464e+0)	2.5000e+0 (2.204e+0)
f_7	8.1742e-2 − (2.9263e-2)	8.6616e-1 − (4.3408e-1)	1.1051e+0 − (6.6114e-1)	7.1672e-2 − (3.4086e-2)	4.8419e+1 − (2.8474e+1)	1.1136e-1 − (5.2390e-2)	6.3319e-2 − (1.6838e-2)	4.9230e-2 − (2.2971e-2)	1.7558e-2 (5.800e-3)
f_8	-7.6960e+3 ≈ (4.2092e+2)	-6.4301e+3 − (7.3448e+2)	-5.2233e+3 − (4.2305e+2)	-7.6329e+3 ≈ (5.6710e+2)	-7.4567e+3 ≈ (6.5244e+2)	-7.5311e+3 ≈ (2.9851e+2)	-5.5305e+3 − (2.9295e+2)	-7.1206e+3 ≈ (2.0444e+2)	-7.3751e+3 (2.951e+2)
f_9	7.0727e+1 − (1.4655e+1)	1.4357e+2 − (1.9674e+1)	2.6278e+2 − (1.3668e+1)	2.2499e+2 − (2.5540e+1)	3.0943e+2 − (1.3695e+1)	1.0129e+2 − (1.1932e+1)	2.1434e+2 − (1.3079e+1)	8.0586e+1 − (1.7108e+1)	4.1289e+1 (1.364e+1)
f_{10}	2.9250e+0 − (1.0910e-1)	1.1749e+1 − (5.8613e-1)	1.3413e+1 − (8.0644e-1)	4.3282e+0 − (4.3019e-1)	1.5657e+1 − (1.5809e+0)	7.3892e+0 − (9.1288e-1)	3.0076e+0 − (3.2128e-1)	2.3761e+0 − (5.8902e-1)	9.6926e-1 (6.627e-1)
f_{11}	1.1551e+0 − (2.4035e-2)	2.9789e+1 − (8.3839e+0)	4.5249e+1 − (8.3656e+0)	1.9904e+0 − (2.2665e-1)	1.1321e+2 − (2.2152e+1)	1.2782e+1 − (4.1098e+0)	1.1978e+0 − (7.2707e-2)	9.4447e-1 − (7.3867e-2)	7.6236e-1 (1.343e-1)
f_{12}	3.5756e+1 − (1.7714e+1)	1.5007e+3 − (1.6007e+3)	1.9844e+5 − (1.8928e+5)	3.7031e+1 − (6.1593e+0)	1.2756e+6 − (1.1497e+6)	1.3627e+2 − (6.7341e+1)	3.4071e+1 − (5.6141e+0)	7.2069e+1 − (1.6143e+1)	1.4346e+0 (1.133e+0)
f_{13}	4.1967e+0 − (3.5806e+0)	6.7210e+5 − (4.4019e+5)	3.8989e+6 − (2.7548e+6)	1.0194e+1 − (3.2998e+0)	9.2196e+6 − (7.2385e+6)	7.5595e+3 − (2.0845e+4)	5.1567e+0 − (1.6990e+0)	6.4904e+0 − (1.5676e+0)	1.4557e-1 (6.230e-2)
+/ − / ≈	0/12/1	0/13/0	0/13/0	0/12/1	0/12/1	0/11/2	0/12/1	1/9/3	

'+', '−' and '≈' indicate that the result is significantly better, significantly worse, and statistically similar to that obtained by AutoV.

Table 9: Minimum objective values obtained by nine algorithms on the CEC'2013 large-scale benchmark problems with about 1000 decision variables. Best results are highlighted

CEC'2013 large-scale problems[42]	GA[6]	PSO[10]	DE[7]	CMA-ES[8]	FEP[9]	CSO[37]	SHADE[38]	IMODE[39]	AutoV
f_1	1.1824e+9 − (1.5088e+8)	1.7274e+11 − (8.2047e+9)	1.1820e+11 − (5.7453e+9)	1.3230e+10 − (5.9476e+8)	6.2225e+10 − (8.8355e+9)	1.6506e+11 − (6.4000e+9)	9.2604e+8 − (9.2525e+7)	1.5672e+10 − (1.8277e+9)	7.5437e+8 (3.599e+7)
f_2	7.0854e+3 + (5.5474e+2)	4.8900e+4 − (1.6515e+3)	4.2656e+4 − (9.8044e+2)	1.1326e+4 + (2.6942e+2)	8.4194e+4 − (5.6859e+2)	4.4704e+4 − (7.5103e+2)	1.8598e+4 − (1.1309e+3)	2.4830e+4 − (6.9359e+2)	1.2736e+4 (3.996e+2)
f_3	1.9790e+1 − (1.0792e-1)	2.1096e+1 − (3.6567e-2)	2.1011e+1 − (6.4041e-3)	2.1334e+1 − (2.0615e-2)	2.1457e+1 − (8.6678e-3)	2.0977e+1 − (1.9539e-2)	1.7116e+1 − (2.1804e-1)	2.0013e+1 − (1.6968e-2)	7.6972e+0 (2.709e-1)
f_4	7.0214e+11 − (3.6361e+11)	3.8264e+12 − (1.0018e+12)	2.8759e+11 − (4.4452e+10)	1.8046e+12 − (2.2407e+11)	7.4290e+11 − (4.1395e+11)	2.0687e+12 − (5.9504e+11)	4.1680e+10 + (7.7120e+9)	3.5605e+11 − (3.1439e+10)	1.7434e+11 (3.356e+10)
f_5	8.0929e+6 − (1.1105e+6)	3.3051e+7 − (3.3554e+6)	1.6648e+7 − (1.2160e+6)	1.1814e+7 − (3.5863e+5)	2.0252e+7 − (1.9684e+6)	2.7747e+7 − (2.3903e+6)	6.6297e+6 − (8.5226e+5)	1.0851e+7 − (6.3028e+5)	5.5157e+6 (7.817e+5)
f_6	7.6705e+5 − (4.2587e+4)	1.0156e+6 − (6.2467e+3)	9.6778e+5 − (1.9218e+4)	1.0725e+6 − (1.4402e+3)	7.7643e+5 − (6.3779e+4)	9.9765e+5 − (5.1452e+3)	1.0802e+5 − (1.1582e+4)	2.7032e+5 − (4.2843e+4)	8.3894e+4 (9.288e+3)
f_7	7.8310e+9 − (2.9908e+9)	4.9258e+13 − (2.3576e+13)	2.3870e+12 − (7.2173e+11)	1.1084e+10 − (9.4838e+9)	2.8185e+11 − (1.4497e+11)	2.2211e+13 − (8.4947e+12)	8.9123e+8 ≈ (1.9320e+8)	1.3523e+10 − (1.6130e+9)	8.4166e+8 (7.997e+7)
f_8	2.9461e+16 − (1.4136e+16)	1.6998e+17 − (9.1594e+16)	1.2728e+13 + (1.9676e+12)	4.7204e+16 − (6.0409e+15)	6.1725e+15 ≈ (3.8102e+15)	3.2323e+16 − (3.0414e+16)	3.7727e+13 + (3.0187e+13)	5.4599e+15 ≈ (2.5604e+15)	4.4372e+15 (1.770e+15)
f_9	7.5728e+8 − (1.2373e+8)	2.5298e+9 − (2.0469e+8)	1.2725e+9 − (6.3827e+7)	8.4502e+8 − (4.5495e+7)	1.6577e+9 − (1.3394e+8)	2.0244e+9 − (2.1854e+8)	5.8016e+8 − (1.1320e+8)	7.7668e+8 − (4.1594e+7)	4.4601e+8 (8.503e+7)
f_{10}	4.4221e+7 − (1.3077e+7)	8.7041e+7 − (2.6524e+6)	2.1046e+6 − (6.4391e+5)	9.6156e+7 − (4.6215e+5)	1.7440e+7 − (3.3440e+6)	7.6458e+7 − (5.2418e+6)	1.2506e+6 − (1.4290e+4)	7.8125e+5 ≈ (2.8894e+5)	4.8728e+5 (4.870e+5)
f_{11}	7.4972e+11 − (4.5447e+11)	3.0398e+15 − (1.4680e+15)	1.1061e+14 − (9.8496e+13)	4.0919e+11 − (1.0871e+11)	1.1334e+13 − (5.1094e+12)	8.9529e+14 − (3.6702e+14)	8.7764e+9 + (4.5005e+9)	4.4608e+11 − (1.6708e+11)	1.6943e+11 (1.144e+11)
f_{12}	2.0599e+10 − (3.9186e+9)	1.9723e+12 − (8.1522e+10)	1.5925e+12 − (3.3820e+10)	1.3548e+10 − (1.3457e+10)	2.6665e+12 − (1.0994e+11)	1.6723e+12 − (2.5065e+10)	3.3481e+10 − (5.8270e+9)	6.4235e+11 − (1.9268e+10)	7.6900e+8 (2.144e+7)
f_{13}	4.0277e+10 − (1.4024e+10)	1.9190e+15 − (8.0609e+14)	5.4725e+13 − (1.7016e+13)	8.3330e+10 − (1.4211e+10)	3.0193e+12 − (1.6034e+12)	1.0082e+15 − (6.2874e+14)	1.0448e+10 + (2.3651e+9)	6.6059e+10 − (9.5957e+9)	1.7051e+10 (4.590e+9)
f_{14}	1.0031e+12 − (3.8285e+11)	4.5075e+15 − (3.3588e+15)	1.2185e+14 − (7.0856e+13)	9.6175e+11 − (4.0377e+11)	1.2889e+13 − (4.1028e+12)	1.1075e+15 − (4.3764e+14)	7.7424e+10 + (1.3029e+10)	7.2323e+11 − (8.4437e+10)	2.3734e+11 (7.915e+10)
f_{15}	5.7586e+7 − (1.5169e+7)	1.8722e+15 − (2.4943e+14)	3.4395e+14 − (4.9325e+13)	9.9936e+8 − (1.1950e+9)	8.8170e+13 − (2.5655e+13)	8.7705e+14 − (1.2736e+14)	3.0449e+7 ≈ (4.6006e+6)	6.1844e+12 − (1.4191e+12)	2.7840e+7 (2.630e+6)
+/ − / ≈	1/14/0	0/15/0	1/14/0	1/14/0	0/14/1	0/15/0	5/8/2	0/13/2	

'+', '−' and '≈' indicate that the result is significantly better, significantly worse, and statistically similar to that obtained by AutoV.

metaheuristics, averaged over 30 runs. It can be observed from the statistical results that the proposed AutoV also exhibits better overall performance than the others on

the large-scale benchmark problems, achieving the best results on 9 out of 15 problems. It is noteworthy that although SHADE and IMODE suggest many complex search strategies for the combination of multiple operators and adaptation of parameters, they are still underperformed by AutoV that only contains a simple operator designed automatically. Therefore, the proposed AutoV offers bright prospects to the design of metaheuristics, which can potentially replace the laborious manual design process.

6 Conclusions

To reduce the human expertise in designing metaheuristics, this paper has analyzed the importance of translation, scale, and rotation invariance to the robustness of operators, and deduced the generic form of translation, scale, rotation invariant operators. Based on the deduced generic form, this paper has proposed a principled approach to search for high-performance operators automatically. In contrast to the automated design approaches based on existing metaheuristics, the proposed approach does not rely on any existing techniques, and can obtain competitive performance over some state-of-the-art metaheuristics on complex and large-scale optimization problems.

The experimental results have demonstrated the effectiveness and potential of the automated design of variation operators, and further research on this topic is highly desirable. Firstly, it is reasonable to search for high-performance operators based on more complex functions (e.g., include more parents and consider the update of velocity), where more effective operators are expected to be found. Secondly, since the proposed approach searches for operators according to their performance on a benchmark problem, it is reasonable to adopt more representative and practical problems to find high-performance operators for general problems or specific types of problems. Thirdly, it is interesting to develop novel approaches for automatically designing selection strategies for metaheuristics. Fourthly, since deep reinforcement learning has been applied in metaheuristics to tune the parameters of operators [43, 44], it can be adopted by the proposed approach to find high-performance operators before and during the optimization of specific problems.

References

[1] M. R. Alam, K. S. Lee, M. Rahman and Y. F. Zhang, "Process planning optimization for the manufacture of injection moulds using a genetic algorithm," *International Journal of Computer Integrated Manufacturing*, vol. 16, no. 3, pp. 181–191, 2003.

[2] J. Y. Potvin and S. Bengio, "The vehicle routing problem with time windows part II: Genetic search," *INFORMS Journal on Computing*, vol. 8, no. 2, pp. 165–172, 1996.

[3] Y. Tian, X. Su, Y. Su and X. Zhang, "EMODMI: A multi-objective optimization based method to identify disease modules," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2020.

[4] K. J. Oh, T. Y. Kim and S. Min, "Using genetic algorithm to support portfolio optimization for index fund management," *Expert Systems with Applications*, vol. 28, pp. 371–379, 2005.

[5] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.

[6] J. H. Holland, *Adaptation in Natural and Artificial Systems*. MIT Press, 1992.

[7] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[8] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[9] X. Yao, Y. Liu and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[10] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, Nagoya, Japan, pp. 39–43, 1995.

[11] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1155–1173, 2008.

[12] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Erciyes University, Tech. Rep. tr06, 2005.

[13] J. Galletly, *Evolutionary algorithms in theory and practice: Evolution strategies, evolutionary programming, genetic algorithms*. Kybernetes, 1998.

[14] T. Okabe, Y. Jin and B. Sendhoff, "Theoretical comparisons of search dynamics of genetic algorithms and evolution strategies," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, 2005, pp. 382–389.

[15] S. Li, Y. Li and Y. Lin, *Intelligent optimization algorithms and emergent computation*. Beijing: Tsinghua University Press, 2019.

[16] A. E. Eiben and C. A. Schippers, "On evolutionary exploration and exploitation," *Fundamenta Informaticae*, vol. 35, pp. 1–16, 1998.

[17] Y. Su, N. Guo, Y. Tian and X. Zhang, "A non-revisiting genetic algorithm based on a novel binary space partition tree," *Information Sciences*, vol. 512, pp. 661–674, 2020.

[18] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.

[19] T. Rodemann, "Industrial portfolio management for many-objective optimization algorithms," in *Proceedings of the 2018 IEEE Congress on Evolutionary Computation*, Rio, Brazil, 2018.

[20] C. A. Coello Coello and M. S. Lechuga, "MOPSO: A proposal for multiple objective particle swarm optimization," in *Proceedings of the 2002 IEEE Congress on Evolutionary Computation*, Hawaii, USA, vol. 2, pp. 1051–1056, 2002.

[21] M. Jebalia, A. Auger and N. Hansen, "Log-linear convergence and divergence of the scale-invariant (1+1)-ES in noisy environments," *Algorithmica*, vol. 59, no. 3, pp. 425–460, 2011.

[22] F. Caraffini and F. Neri, "A study on rotation invariance in differential evolution," *Swarm and Evolutionary Computation*, vol. 50, p. 100436, 2019.

[23] G. Karafotias, M. Hoogendoorn and A. E. Eiben, "Parameter control in evolutionary algorithms: Trends and challenges," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 167–187, 2015.

[24] C. Huang, Y. Li and X. Yao, "A survey of automatic parameter tuning methods for metaheuristics," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 201–216, 2020.

[25] P. Kerschke and H. Trautmann, "Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning," *Evolutionary Computation*, vol. 27, no. 1, pp. 99–127, 2019.

[26] Y. Tian, S. Peng, X. Zhang, T. Rodemann, K. C. Tan and Y. Jin, "A recommender system for metaheuristic algorithms for continuous optimization based on deep recurrent neural networks," *IEEE Transactions on Artificial Intelligence*, vol. 1, no. 1, pp. 5–18, 2020.

[27] K. Li, A. Fialho, S. Kwong and Q. Zhang, "Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 114–130, 2013.

[28] L. C. Bezerra, M. López-Ibánez and T. Stützle, "Automatic component-wise design of multiobjective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 403–417, 2016.

[29] L. P. M. Birattari, T. Stützle and K. Varrentrapp, "A racing algorithm for configuring metaheuristics," in *Proceedings of the 2002 Conference on Genetic and Evolutionary Computation*, New York, USA, pp. 11–18, 2002.

[30] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[31] A. Fialho, M. Schoenauer and M. Sebag, "Toward comparison-based adaptive operator selection," in *Proceedings of the 2010 Conference on Genetic and Evolutionary Computation*, Portland, USA, pp. 767–774, 2002.

[32] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, no. 4, pp. 115–148, 1995.

[33] C. Igel, N. Hansen and S. Roth, "Covariance matrix adaptation for multi-objective optimization," *Evolutionary Computation*, vol. 15, no. 1, pp. 1–28, 2007.

[34] N. Hansen, R. Ros, N. Mauny, M. Schoenauer and A. Auger, "Impacts of invariance in search: When CMA-ES and PSO face ill-conditioned and non-separable problems," *Applied Soft Computing*, vol. 11, no. 8, pp. 5755–5769, 2011.

[35] K. A. Hoffmann and S. T. Chiang, *Computational Fluid Dynamics Volume I*. Engineering Education System, 2000.

[36] L. Pan, W. Xu, L. Li, C. He and R. Cheng, "Adaptive simulated binary crossover for rotated multi-objective optimization," *Swarm and Evolutionary Computation*, vol. 60, p. 100759, 2021.

[37] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 191–204, 2014.

[38] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proceedings of the 2013 IEEE Congress on Evolutionary Computation*, Cancun, Mexico, 2013.

[39] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty and M. J. Ryan, "Improved multi-operator differential evolution algorithm for solving unconstrained problems," in *Proceedings of the 2020 IEEE Congress on Evolutionary Computation*, Glasgow, U. K., 2020.

[40] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," *Computer Science and Informatics*, vol. 26, no. 4, pp. 30–45, 1996.

[41] J. Derrac, S. Garcia, D. Molina and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[42] X. Li, K. Tang, M. N. Omidvar, Z. Yang and K. Qin, "Benchmark functions for the CEC'2013 special session and competition on large-scale global optimization," *Gene*, vol. 7, no. 33, 2013.

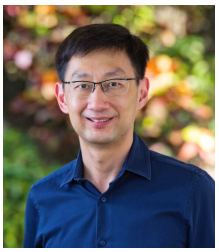
[43] J. Sun, X. Liu, T. Bäck and Z. Xu, "Learning adaptive differential evolution algorithm from optimization experiences by policy gradient," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 666–680, 2021.

[44] H. A. A. Nomer, A. W. Mohamed and A. H. Yousef, "GSK-RL: Adaptive gaining-sharing knowledge algorithm using reinforcement learning," in *Proceedings of the 3rd Novel Intelligent and Leading Emerging Sciences Conference*, Giza, Egypt, 2021.

TIAN Ye was born in Anhui Province in 1991. He received the B.Sc., M.Sc., and Ph.D. degrees from Anhui University, Hefei, China, in 2012, 2015, and 2018, respectively. He is currently an Associate Professor with the Institutes of Physical Science and Information Technology, Anhui University, Hefei, China. His current research interests include evolutionary computation and its applications. (Email: field910921@gmail.com)

ZHANG Xingyi (corresponding author) was born in Anhui Province in 1982. He received the B.Sc. degree from Fuyang Normal College, Fuyang, China, in 2003, and the M.Sc. and Ph.D. degrees from Huazhong University of Science and Technology, Wuhan, China, in 2006 and 2009, respectively. He is currently a Professor with the School of Artificial Intelligence, Anhui University, Hefei, China. His current research interests include unconventional models and algorithms of computation, evolutionary multi-objective optimization, and logistic scheduling. (Email: xyzhanghust@gmail.com)

HE Cheng was born in Hubei Province in 1989. He received the B.Eng. degree from the Wuhan University of Science and Technology, Wuhan, China, in 2012, and the Ph.D. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2018. He is currently an Associate Professor with the School of Electrical and Electronic Engineering, Huazhong University of Science and Technology, Wuhan, China. His current research interests include model-based evolutionary algorithms, multiobjective optimization, large-scale optimization, deep learning, and their applications. (Email: chenghehust@gmail.com)



TAN Kay Chen received the B.Eng. (First Class Hons.) and Ph.D. degrees from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively. He is currently a Chair Professor of the Department of Computing at the Hong Kong Polytechnic University, Hong Kong SAR, China. (Email: kctan@polyu.edu.hk)

JIN Yaochu was born in Jiangsu Province in 1966. He received the B.Sc., M.Sc., and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 1988, 1991, and 1996, respectively, and the Dr.-Ing. degree from Ruhr University Bochum, Germany, in 2001. He is currently an Alexander von Humboldt Professor for Artificial Intelligence, Chair of Nature Inspired Computing and Engineering, Faculty of Technology, Bielefeld University, Germany. He is also a Distinguished Chair, Professor in Computational Intelligence, Department of Computer Science, University of Surrey, Guildford, U.K. (Email: yaochu.jin@uni-bielefeld.de)