A Mixed Strategy for Evolutionary Programming Based on Local Fitness Landscape

Liang, Shen; He, Jun

DOI:
10.1109/CEC.2010.5586414

Publication date:
2010

Citation for published version (APA):
Liang, S., & He, J. (2010). *A Mixed Strategy for Evolutionary Programming Based on Local Fitness Landscape*. 350. Paper presented at IEEE Congress on Evolutionary Computation (CEC), Barcelona, Spain. https://doi.org/10.1109/CEC.2010.5586414

A Mixed Strategy for Evolutionary Programming Based on Local Fitness Landscape

Liang Shen and Jun He

Abstract— **The performance of Evolutionary Programming (EP) is affected by many factors (e.g. mutation operators and selection strategies). Although the conventional approach with Gaussian mutation operator may be efficient, the initial scale of the whole population can be very large. This may lead to the conventional EP taking too long to reach convergence. To combat this problem, EP has been modified in various ways. In particular, modifications of the mutation operator may significantly improve the performance of EP. However, operators are only efficient within certain fitness landscapes. The mixed strategies have therefore been proposed in order to combine the advantages of different operators. The design of a mixed strategy is currently based on the performance of applying individual operators. Little is directly relevant to the information of local fitness landscapes. This paper presents a modified mixed strategy, which automatically adapts to local fitness landscapes, and implements a training procedure to choose an optimal mixed strategy for a given typical fitness landscape. The proposed algorithm is tested on a suite of 23 benchmark functions, demonstrating the advantages of this work in that it is less likely to be stuck in local optima and has a faster and better convergence.**

I. Introduction

Evolutionary programming (EP) is a branch, alongside other notable research areas such as genetic algorithms and evolution strategy, of evolutionary computation that stems from natural biological evolution [1]. EP operates on the basis of populations. The objective is not only to find suitable adjustments to the current population and hence the solution, but also to perform the process efficiently. However, a parameter setting that was optimal at the beginning of an EP-run may become unsuitable during the evolutionary process. Thus, it is desirable to automatically modify the control parameters during the run of an EP. Control parameters can be of various forms, ranging from mutation rates, recombination probabilities, and population size to selection operators. In light of this, self-adaptation techniques [2] have been introduced to implement such parameter control. The approach has a bias on the distribution towards appropriate directions of the search space, thereby maintaining sufficient diversity among individuals in order to enable further ability of evolution.

As a key element in EP, mutation operators have attracted significant attention in research. The original mutation operator is typically Gaussian, which usually lacks robustness when applied to multi-modal functions. A substituent of Gaussian mutation in fast evolutionary programming (FEP) as reported in [3] entails better performance regarding many multivariate functions. However, it is less efficient on some unimodal functions. By generalizing FEP further using mutation based on the Lévy probability distribution, further improvement can be achieved [4].

Unfortunately, no single algorithm or operator is the best on average for all problems [5], be they self-adaptation or not. An approach for enhancing the conventional EP that uses a single mutation operator is to apply different mutation operators simultaneously and integrate their advantages together. Such a strategy is called a mixed mutation strategy (borrowing the concept from game theory [6]). The employment of a mixed strategy stems from the need to explore a unified approach for maximizing the ability of various self-adaptive strategies, while assuming no prior knowledge of the problems at hand.

There are different ways to design a mixed strategy. For example, an early implementation is a linear combination of Gaussian and Cauchy distributions [7]. An alternative approach is the improved fast EP (IFEP) [3], [4] which works by: 1) each individual of a population implementing Cauchy and Gaussian mutations simultaneously and generating two offspring; and 2) the better one being chosen to construct the next generation. These two approaches are simple in implementation. Reinforcement learning theory may also be used to learn individual mutation operators [8]. Progress has recently been made in an effort to adjust mutation strategies [6], [9], [10].

In previous studies [9], [11], the design of a mixed strategy mainly utilizes the reward of each operator (i.e. a operator which produces a higher fitness will receive a better reward). Little existing work is directly relevant to the information of local fitness landscapes. However, the performance of each mutation operator is strongly linked to the fitness landscape, so it is important to deal with the local fitness landscape where an population is located. In this paper, a novel mixed strategy is proposed in order for the strategy to adapt to the given fitness landscape.

The rest of the paper is organized as follows. Section II outlines the general procedure of conventional EP using a single mutation operator. Section III-A focuses on LM-SEP. It first introduces a measure about the local fitness landscape on the multi-modality. Then it describes the new mixed strategy with respect to this measure. After that, it discusses the training procedures for learning local fitness landscapes. Section IV reports the experimental results on the benchmark problems. Section V concludes the paper, with further research pointed out.

Liang Shen (email: lls08@aber.ac.uk) and Jun He (email: jqh@aber.ac.uk) are with the Department of Computer Science, Aberystwyth University, Wales, UK.

978-1-4244-8126-2/10/$26.00

II. Basic Operations of Evolutionary Programming

In this paper EP is used to find a minimum $\vec{x}_{\min}$ of a continuous function $f(\vec{x})$, that is,

$$f(\vec{x}_{\min}) \leq f(\vec{x}), \quad \vec{x} \in D, \tag{1}$$

where D is a hypercube in $\mathbb{R}^n$, n is the dimension. Conventional EP which uses a single mutation operator can be described as follows [3]:

1) **Initialization:** Generate an initial population consisting of μ individuals at random. Each individual is represented by a set of real vectors $(\vec{x}_i, \vec{\sigma}_i)$, for , $i = 1, \cdots, \mu$

$$\vec{x}_i = (x_i(1), x_i(2), \cdots, x_i(n)),$$
$$\vec{\sigma}_i = (\sigma_i(1), \sigma_i(2), \cdots, \sigma_i(n)).$$

2) **Mutation:** For each parent $(\vec{x}_i^{(t)}, \vec{\sigma}_i^{(t)})$ (where t represents generation), create an offspring $(\vec{x}_i', \vec{\sigma}_i')$ as follows: for $j = 1, \cdots, n$,

$$\begin{aligned} \sigma_i'(j) &= \sigma_i^{(t)}(j) \exp\{\tau N(0,1) + \tau' N_j(0,1)\}, \\ x_i'(j) &= x_i^{(t)}(j) + \sigma_i^{(t+1)}(j) X_j, \end{aligned} \tag{2}$$

where $N(0,1)$ stands for a Gaussian random variable generated for a given i, $N_j(0,1)$ is a Gaussian random variable generated for each j, and X_j is a random variable generated for each j. Controlling parameters τ and τ' are chosen as the same as in [3].

3) **Fitness Evaluation:** For μ parents and their μ offspring, calculate their fitness values $f_1, f_2, \cdots, f_{2\mu}$.
4) **Selection:** Define and initialize a winning function for every individual in parent and offspring population as $w_i = 0$, $i = 1, 2, \cdots, 2\mu$. For each individual i, select one fitness function, say f_j and compare the two fitness functions. If f_i is less than f_j, then let $w_i = w_i + 1$. Perform this procedure q times for each individual. Select μ individuals that have the largest winning values to be the parents of the next generation.
5) Repeat steps 2-4, until the stopping criteria are satisfied.

To avoid the step size σ falling too low, a lower bound $\sigma_{\min}$ should be put on σ [12]. So a revised scheme of updating σ is given by:

$$\sigma_i'(j) = (\sigma_{\min} + \sigma_i^{(t)}(j)) \exp\{\tau N(0,1) + \tau' X_j\}.$$

where $\sigma_{\min} > 0$ is the minimum value of step size σ.

III. A Novel Mixed Strategy

This section describes a novel mixed strategy that is developed in order to improve the conventional EP by two major techniques:

1) A mixed mutation strategy, based on the observation that local fitness landscapes form a key factor in the determination of the behavior of mutations in EP.
2) A training procedure, where several typical learning functions are introduced (as the training dataset) in order to determine the preferable probability distribution of mixed mutation operators with respect to different types of local fitness landscape.

These two tasks will be addressed below, which are then combined to deal with a set of target functions.

A. Mixed strategy adapting to local fitness landscape

In EP, a mutation operator is determined by the random variable X_j given in Eq. (2), which satisfies the probability distribution function F_s. A mutation operator is denoted by s. Currently, the set of mutation operators consists of Cauchy, Gaussian, Lévy and other probability distributions, and the set is denoted by $S = \{s_1, \cdots, s_L\}$.

With this notion in mind, a mixed strategy based on the probability distribution can be developed. The mixed strategy can be described as follows: at each generation, an individual chooses one mutation operator s from its strategy set based on a selection probability distribution $\rho(s)$. A mixed strategy distribution is determined by $\pi = (\rho(s_1), \cdots, \rho(s_L))$.

The key problem in the mixed strategy is to find out a good, if possible an optimal, probability distribution $(\rho(s_1), \cdots, \rho(s_L))$ for every individual. This distribution is dynamic, which changes over generations. The problem can be formalized as follows: Given the t-th generation population, decide a probability distribution of $\pi = (\rho(s_1), \cdots, \rho(s_L))$ which maximizes the drift towards the global optima, i.e.,

$$\max_{\pi} \{d(\vec{x}^{(t)}, \vec{y});\ \vec{y} \in S_{\min})\}, \tag{3}$$

where $S_{\min}$ is the global optimal set, and $d(\vec{x}, \vec{y})$ is the Euclidean distance.

In theory, such an optimal mixed strategy π always exists, but in practice it is impossible to find out the optimal strategy π since the optimal set $S_{\min}$ is unknown.

Instead, the mixed strategy is designed on the basis of following assumption that the mixed strategy should adapt to local fitness landscape. In this paper the following principle is taken from previous experiments [6]: 1) If the local fitness landscape looks like uni-modal landscape, Gaussian mutation should be applied with a higher probability; 2) if the local fitness landscape looks like a multi-modal fitness landscape, then Cauchy mutation is applied with a higher probability.

There are two tasks in designing the above mixed strategy: (1) given an individual $\vec{x}$ in the real space $\mathbb{R}^n$, determine what type of local fitness landscape it looks like; (2) based on the characteristics of local fitness landscape, assign a probability distribution for the mixed strategy.

Consider the first task. Given an individual $\vec{x}$, it is difficult to give the precise characteristics of local fitness landscape and the computation cost will be very heavy. Instead it will be better to seek a simplified approach. Since each individual is among a population, the population forms an observation of the local fitness landscape. A simple feature of the local fitness landscape then is drawn from the observation. Sorting other individuals in the population based on their distances from the best individual in the population, then check how the fitness of each changes over the distance. If the fitness is

increasing with the distance, then the local fitness landscape is like a uni-modal landscape; otherwise, it belongs to a multi-modal landscape. A simple procedure to implement this is given as follows. For a population $(x_1, \cdots, x_\mu)$,

1) Find out the best individual among the population, mark it with x_{best}. Then calculate the distance between each individual $x_i (i = 1, \cdots, \mu)$ and x_{best} as follows:

$$d_i = \sum_{j=1}^{n} \mid x_{i_j} - x_{best_j} \mid . \tag{4}$$

2) Sort the individuals based on the distance value, resulting in the following in ascending order: $k_1, \cdots, k_\mu$
3) Calculate the measure of the individual on the local fitness landscape. Denote the measure value by χ. Assume the value to be 0 initially, then the value will be increased by 1 if $f_{k_{i+1}} \leq f_{k_i}$. Normalize the value as:

$$\varphi = \frac{\chi}{\mu}. \tag{5}$$

The second task is based on learning. Given several typical fitness landscapes, calculate the performance of different mixed strategy on these fitness landscapes and find the best mixed strategy for each landscape feature φ. As local fitness landscape is actual a fuzzy concept, the feature φ can be regarded as the roughness of observed fitness landscape.

In this paper, only two mutation operators are used, i.e., Gaussian and Cauchy mutation, The performance of these two mutation operators is well known [3], [6]; they behave just in an exactly opposite way. Therefore, to determine the mixed strategy $\pi = (\rho(s_{Cauchy}), \rho(s_{Gaussian}))$, a straightforward approach is used: The probability of using Cauchy mutation can be treated to be numerically equal to the feature φ. Likewise, the probability of Gaussian mutation equals $(1 - \varphi)$.

$$\begin{cases} \rho(s_{Cauchy}) & = \varphi \\ \rho(s_{Gaussian}) & = 1 - \varphi \end{cases} \quad \varphi \in [0, 1]. \tag{6}$$

Hence, for mixed strategy including only Cauchy and Gaussian mutation, the probability distribution is

$$\pi = (\varphi, (1 - \varphi)), \quad \varphi \in [0, 1]. \tag{7}$$

This is reasonable because: if the value of $\varphi = 0$, then local fitness landscape is more like a unimodal landscape, thus it is better to use Gaussian mutation only; if the value of $\varphi = 1$, then local fitness landscape is very rough, it may be good for applying Cauchy mutation only. As the value of φ increases, the probability of apply Cauchy mutations should be increased.

B. Training mechanism for learning local fitness landscape

The above study shows the design of a novel mixed mutation strategy based on local fitness landscapes. Its implementations involves the fluctuation of the proportion for each mutation operator to be applied in every generation. As the performance of Cauchy and Gaussian mutation is well-known (with them simply behaving in an opposite way), the mixed strategy π can be determined by Eq. (6). However, this implies that an existing mixed strategy corresponding to a certain local fitness landscape has already been determined via human intervention. It also makes the algorithm resistant to the use of any novel mutation operator without sufficient prior knowledge about the problems at hand. In view of this, it is desirable if the mixed strategy S_x regarding to the given local fitness landscape φ_x can be self-determined and generalized to similar cases.

This can be accomplished by introducing a training procedure prior to running the algorithm on target functions. The task of finding the global minimum in numerical function optimization is herein implemented by learning rules based on experience gained from prior performance [13]. In particular, a suite of functions are considered as the training examples. The set of training functions $\{f_1, \cdots, f_\gamma\}$ is chosen to be the representatives of different local fitness landscapes, e.g., unimodal functions, multimodal functions with many local minima, and multimodal functions with only a few local minima.

Features of local fitness landscapes φ, as well as the corresponding mixed strategy, are required to construct the actual training data. They can be obtained by taking the advantage of the algorithm presented in Section III-A. Note that as the algorithm is used to test the entire process of function optimization, it may be performed all along until certain performance criteria are satisfied or the maximum execution time is reached. Under normal circumstances, the algorithm will terminate at a plateau state, suggesting it could not find any better result. Also, prior to reaching this state, it usually will have experienced an entire operation region involving a large number of intermediate states. These intermediate states may exhibit a variety of fitness landscapes that may vary in terms of the lapse of time. Hence, if the algorithm runs on a multimodal function, after running a large number of generations, individuals in the population may shrank to a limit region in the vicinity of the global optimal, in favor of similar results. Since differences between them will be considerably small, the underlying local fitness landscape will look like a unimodal. In order to ensure that all individuals are located uniformly and randomly in the entire domain of training functions, the algorithm is slightly modified. This is achieved by running a relative small number t_T of generations by which the results of each training function are averaged.

According to Eq. (5), feature φ can be set to a different value $\{\varphi_1, \cdots, \varphi_n\}$, based on a set of training functions $\{f_1, \cdots, f_n\}$. Thus, the probability distribution of the mutation operator is needed. For a mixed strategy involving only Cauchy and Gaussian mutation, it can be calculated using Eq. (6). To be consistent with φ, the probability distribution is also averaged by t_T as follows:

$$\begin{cases} \rho(s_{Cauchy}) & = \sum_{i=1}^{t_T} \frac{\varphi_i}{t_T} \\ \rho(s_{Gaussian}) & = \sum_{i=1}^{t_T} \frac{1-\varphi_i}{t_T} \end{cases} \quad \varphi_i \in [0,1]. \tag{8}$$

While $\varphi, \cdots, \varphi_k$ are every single value obtained from every generation among t_T ones.

As γ individual functions are chosen to form the training examples, γ pairs of features and probability distribution are calculated. The set of features $\{\varphi_1, \cdots, \varphi_\gamma\}$ and probability distributions $(\rho(s_1), \cdots, \rho(s_L))$ have a one-to-one correspondence. It is reasonable to assume that a target probability distribution, given a feature, can be learned from the underlying correspondence by taking advantage of the existing ones, $\{\varphi_1, \cdots, \varphi_\gamma\}$.

In order to implement the learning of the correspondence between feature φ_x and the target, π_x, a distance-based approach is utilized to approximate the best π_x. All training data as well as π_x are considered as points in a line in terms of the values of feature φ. Two points with a relatively low distance are regarded as neighbors. Given a target feature φ_x whose value is in the interval $[0, 1]$, one instance φ_a among training data $\{\varphi_1, \cdots, \varphi_\gamma\}$ can be found as its nearest neighbor. Therefore, this approach has an intuitive appealing, treating π_a that is associated with φ_a as the required approximation. The probabilities of each mutation operator in the required π_x are

$$\rho_x(s) = \rho_a(s). \tag{9}$$

It is reasonable to adopt Eq. (9) when there is only one point in the vicinity of target feature π_x. However, it is possible that the target feature is given in the location where the distances

$$d(\varphi_x, \varphi_i) = \mid \varphi_x - \varphi_i \mid . \tag{10}$$

from its two neighbors are nearly the same. Having taken notice of this, the approximation above is modified so that the two nearest neighbors of the target feature (one on each side) are both taken into account. The contribution of each neighbor is weighted according to its distance to the query point φ_x in order to place a greater weight onto closer neighbors. Denoting the neighbor with a smaller value as φ_a and the other as φ_b, the relative placement factor is defined by

$$\lambda = \frac{\varphi_x - \varphi_a}{\varphi_b - \varphi_a}, \ \ \lambda \in [0, 1]. \tag{11}$$

The probability of each mutation operator of target π_x is then considered as the linear combination of its two neighbors:

$$\rho_x(s) = (1-\lambda)\rho_a(s) + \lambda\rho_b(s), \ \ \lambda \in [0, 1]. \tag{12}$$

Note that if the target feature φ happens to take a value close to 0 or 1, Eq. (9) can then be used. In this case, it is obvious that Eq. (12) is not applicable because there is only one existing neighbor.

In summary, the calculation of required π is carried out as follows:

Distance Rule:

1) Given a target feature φ_x, identify its two nearest points φ_a and φ_b among γ training features.
2) Examine the value of φ_a and that of φ_b relative to φ_x. If they are both larger or less than φ_x, then target distribution π is generated as stated in Eq. (9). Otherwise, Eq. (12) is adopted.

C. Evolutionary Programming using a Mixed Strategy

With the aid of aforementioned training procedure, the mixed strategy can be approximately generated automatically in relation to previously unknown local fitness landscapes. The details of this new mixed strategy-based evolutionary programming algorithm are given as follows:

Training: Before applying the mixed strategy, γ functions are employed as training examples so as to generate a set of correspondence between feature φ and probability distribution π of mixed strategy.

T1: **Initialization:** An initial population is generated consisting of μ individuals at random, each of which is represented by two real vectors $\vec{x}_i^{(0)}$ and $\vec{\sigma}_i^{(0)}$ ($i \in 1, 2, \cdots, \mu$). Each $\vec{x}_i^{(0)}$ is a random point in the search space, and each $\vec{\sigma}_i^{(0)}$ is a vector of the coordinate deviations. Both vectors have n real-valued components: for $i = 1, \cdots, \mu$.

$$\vec{x}_i^{(0)} = (x_i^{(0)}(1), x_i^{(0)}(2), \cdots, x_i^{(0)}(n))$$
$$\vec{\sigma}_i^{(0)} = (\sigma_i^{(0)}(1), \sigma_i^{(0)}(2), \cdots, \sigma_i^{(0)}(n))$$

For all individuals, their mixed strategy is taken to be the same one, i.e. $\pi = (\rho^{(0)}(1), \rho^{(0)}(2),)$, where $\rho^{(0)}(1), \rho^{(0)}(2)$ represent the probabilities of choosing Gaussian and Cauchy mutation operators respectively.

T2: **Feature Calculation:** For each individual i in the population, the feature of local fitness landscape can be determined as follows: given a population $(x_1, \cdots, x_\mu)$, assume x_1, without losing generality, is the best individual in the population. Calculate the feature value φ of the local fitness landscape given in Eq. (5).

T3: **Adjustment of Mixed Strategy:** Adjust the probability distribution π_m based on the feature value φ according to Eq. (6). Assign φ to the probability of using Cauchy mutation, $(1-\varphi)$ to Gaussian mutation.

T4: **Mutation:** Denote t to be the generation counter. Each individual i chooses a mutation operator from its strategy set according to its mixed strategy $(\rho^{(t)}(1), \rho^{(t)}(2))$, and uses this strategy to generate a new offspring.

The operator set includes the following two mutation operators. In each description individual parent i is written in the form $(\vec{x}_i^{(t)}, \vec{\sigma}_i^{(t)})$. The corresponding offspring i' is written in the form $(\vec{x}_{i'}^{(t)}, \vec{\sigma}_{i'}^{(t)})$.

Gaussian mutation: Each parent i produces an offspring i' as follows: for $j = 1, 2, \cdots, n$

TABLE I

23 FUNCTIONS USED FOR EXPERIMENTS WHICH ARE CATEGORIZED INTO 3 GROUPS. 7 FUNCTIONS CHOSEN AS TRAINING FUNCTIONS ARE HIGHLIGHTED WITH GREY BACKGROUND. n STANDS FOR THE DIMENSION OF THE FUNCTIONS AND S STANDS FOR THE RANGE OF THE FUNCTIONS.

Test functions	n	Domain	$f_{\min}$
$f_1 = \sum_{i=1}^{30} x_i^2$	30	$[-100, 100]^n$	0
$f_2 = \sum_{i=1}^{30} \mid x_i \mid + \prod_{i=1}^{30} \mid x_i \mid$	30	$[-10, 10]^n$	0
$f_3 = \sum_{i=1}^{30} \left(\sum_{j=1}^{i}\right)^2$	30	$[-100, 100]^n$	0
$f_4 = \max\{\mid x_i \mid, 1 \le i \le n\}$	30	$[-100, 100]^n$	0
$f_5 = \sum_{i=1}^{29} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	30	$[-30, 30]^n$	0
$f_6 = \sum_{i=1}^{30} (\lfloor x_i + 0.5 \rfloor)^2$	30	$[-100, 100]^n$	0
$f_7 = \sum_{i=1}^{30} i x_i^4 + \text{random}[0, 1)$	30	$[-1.28, 1.28]^n$	0
$f_8 = -\sum_{i=1}^{30} \left(x_i \sin(\sqrt{\mid x_i \mid})\right)$	30	$[-500, 500]^n$	-12569.5
$f_9 = \sum_{i=1}^{30} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	30	$[-5.12, 5.12]^n$	0
$f_{10} = -20\exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{30} x_i^2}\right) - \exp\left(\frac{1}{30}\sum_{i=1}^{30}\cos(2\pi x_i)\right) + 20 + e$	30	$[-32, 32]^n$	0
$f_{11} = \frac{1}{4000}\sum_{i=1}^{30} x_i^2 - \prod_{i=1}^{30}\cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]^n$	0
$f_{12} = \frac{\pi}{30}\{10\sin^2(\pi y_1) + \sum_{i=1}^{29}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\}$ $+ \sum_{i=1}^{30} u(x_i, 10, 100, 4), \quad y_i = 1 + \frac{1}{4}(x_i + 1)$	30	$[-50, 50]^n$	0
$f_{13} = 0.1\{10\sin^2(\pi 3x_i) + \sum_{i=1}^{29}(x_i - 1)^2\left[1 + \sin^2(3\pi x_{i+1})\right]$ $+(x_n - 1)^2\left[1 + \sin^2(2\pi x_{30})\right] + \sum_{i=1}^{30} u(x_i, 5, 100, 4)\}$	30	$[-50, 50]^n$	0
$f_{14} = \left[\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right]^{-1}$	2	$[-65.536, 65.536]^n$	0.998
$f_{15} = \sum_{i=1}^{11}\left[a_i - \frac{x_i(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	$[-5, 5]^n$	0.0003075
$f_{16} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]^n$	-1.0316285
$f_{17} = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$	2	$[-5, 10] \times [0, 15]$	0.398
$f_{18} = \left[1 + (x_1 + x_2 + 1)^2(10 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right]$ $\times\left[30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 2x_1^2 + 48x_2 = 36x_1x_2 + 27x_2^2)\right]$	2	$[-2, 2]^n$	3
$f_{19} = -\sum_{i=1}^{3} c_i \exp\left[-\sum_{j=1}^{4} a_{ij}(x_j - p_{ij})\right]$	3	$[0, 1]^n$	-3.86
$f_{20} = \sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})\right]$	6	$[0, 1]^n$	-3.32
$f_{21} = -\sum_{i=1}^{5}\left(\sum_{j=1}^{4}(x_j - a_{ij})^2 + c_i\right)^{-1}$	4	$[0, 10]^n$	-10.15
$f_{22} = -\sum_{i=1}^{7}\left(\sum_{j=1}^{4}(x_j - a_{ij})^2 + c_i\right)^{-1}$	4	$[0, 10]^n$	-10.34
$f_{23} = -\sum_{i=1}^{10}\left(\sum_{j=1}^{4}(x_j - a_{ij})^2 + c_i\right)^{-1}$	4	$[0, 10]^n$	-10.54

$$\sigma_{i'}^{(t)}(j) = \sigma_i^{(t)}(j)\exp\{\tau_a N(0,1) + \tau_b N_j(0,1)\}$$
$$x_{i'}^{(t)}(j) = x_i^{(t)}(j) + \sigma_{i'}^{(t)}(j) N_j(0,1)$$

where $N(0,1)$ stands for a standard Gaussian random variable (fixed for a given i), and $N_j(0,1)$ stands for an independent Gaussian random variable generated for each component j. The control parameter values τ_a and τ_b are chosen as follows:

$$\tau_a = 1/\sqrt{2\mu} \text{ and } \tau_b = 1/\sqrt{2\sqrt{\mu}}.$$

Cauchy Mutation: Each parent i generates an offspring i' as follows: for $j = 1, 2, \cdots, n$

$$\sigma_{i'}^{(t)}(j) = \sigma_i^{(t)}(j)\exp\{\tau_a N(0,1) + \tau_b N_j(0,1)\},$$
$$x_{i'}^{(t)}(j) = x_i^{(t)}(j) + \sigma_{i'}^{(t)}(j)\delta_j$$

where δ_j is a standard Cauchy random variable, which is generated for each component j. The parameters τ_a and τ_b are set to be the values used in the Gaussian mutation.

After mutation, a total of μ new individuals are generated. The offspring population is denoted by $I'(t)$.

T5: **Fitness Evaluation:** Calculate the fitness of individuals in both parent and offspring populations.

T6: **q-Tournament Selection:** For every individual $i \in 1, 2, \cdots, 2\mu$ in the parent and offspring populations, a winning function w_i is initialized to zero. For each

TABLE II
COMPARISON OF MEAN BEST FITNESS BETWEEN LMSEP AND MSEP, LEP, FEP, CEP

	Generations	LMSEP	MSEP	LEP	FEP [3]	CEP [3]
		mean best	mean best	mean best	mean best	mean best
f_1	1500	3.803e-5	6.209e-5	3.048e-2	5.72e-4	1.91e-4
f_2	2000	1.556e-3	8.226e-2	1.232e-2	7.60e-2	2.29e-2
f_4	5000	0.767	0.629	1.053e-3	0.3	2.0
f_6	1500	0	43.8	0	0	577.76
f_7	3000	3.514e-2	3.56e-2	9.197e-3	7.6e-3	1.8e-2
f_9	5000	61.39	63.44	42.064	4.6e-2	89.0
f_{10}	1500	1.956e-3	6.498	1.797e-1	1.76e-2	8.79
f_{11}	2000	5.0e-2	7.768e-2	5.5e-2	2.49e-2	8.13e-2
f_{12}	1500	1.282	9.36e-1	1.3e-2	9.2e-6	1.76
f_{15}	4000	2.247e-4	3.077e-4	4.5e-4	1.8e-2	9.2
f_{16}	100	-1.0316	-1.0316	-1.029	-1.03	-1.03
f_{18}	100	3	3	3.19	3	3
f_{19}	100	-3.863	-3.863	-3.863	-3.86	-3.86
f_{21}	100	-8.744	-8.62	-7.84	-5.50	-6.43
f_{22}	100	-9.585	-9.37	-9.68	-5.73	-7.62
f_{23}	100	-9.483	-9.63	-9.70	-6.41	-8.86

individual i, select another individual j at random and compare fitness $f(i)$ with $f(j)$. If $f_i < f_j$, then the winning function for individual i is increased by one ($w_i = w_i + 1$). This procedure is performed q times for each individual. Based on the winning function, μ individuals from parent and offspring population with highest winning values are selected in the next generation.

T7: **Establishment of Training Dataset:** Steps T2-T6 are repeated for γ times which is the stopping criterion of training procedure. Then, φ and π_m are averaged by following Eq. (8). Once this training dataset is established, no such learning procedures are required to be repeated in testing. This is because the testing procedures only require the use of the mixed strategies resulted from this training process. Therefore, the running cost of training procedure is excluded when evaluating the total cost of testing target functions.

Mixed Strategy for Testing: Upon the completion of training, the EP with the proposed mixed strategy can be run on various target functions, applying it to the established dataset:

P1: The testing procedure involves a general procedure of EP combined with Feature Calculation in preparation for the use of the mixed strategy. Several steps are to be performed in the way that are identical to what is done during the Training Procedure. In particular, the first two steps are identical to T1 and T2.

P2: **Learning Mixed Strategy:** With the knowledge of feature φ_x, the mixed strategy related to it is then determined as follows: Find two similar features in the training dataset using the distance defined in Eq. (11); Consider their relative positions to φ_x, the probability distribution of the mixed strategy is then obtained according to **Distance Rule**.

P3: After this, the steps are also the same to the corresponding ones (T4-T6) of the training procedure.

P4: Steps P1-P2 are repeated until the stopping criterion is satisfied.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

For the purpose of validating the effectiveness of proposed mixed strategy based on local fitness landscape, 23 functions which were used to test FEP in [3] are employed as the training examples. The description of these functions is provided in Table I. They are divided into 3 classes: functions $f_1 - f_7$ are unimodal functions, functions $f_8 - f_{13}$ are multimodal functions with many local minima, and functions $f_{14} - f_{23}$ are multimodal functions with only a few local minima. The parameter setup in the mixed EP is taken to be the same as those in [3]. Population size $\mu = 100$, tournament size $q = 10$, and initial standard deviation is taken as $\sigma = 3.0$. The lower-bound used for them is $\sigma_{\min} = 10^{-5}$ for all functions except f_8 and f_{11}. Since f_8 and f_{11} have larger definition domains than the rest, $\sigma_{\min}$ is taken to be a bigger value 10^{-4}.

To conduct the process of training, some of the functions are chosen as training samples which should consist of all the three types of functions. In this paper, functions f_3, f_5, f_8, f_{13}, f_{14}, f_{17} and f_{20} are chosen as representatives of three classes. The generations t_T of training functions are limited to 5. These 7 training functions are colored in grey background in Table I.

All the other functions are used as actual testing functions.

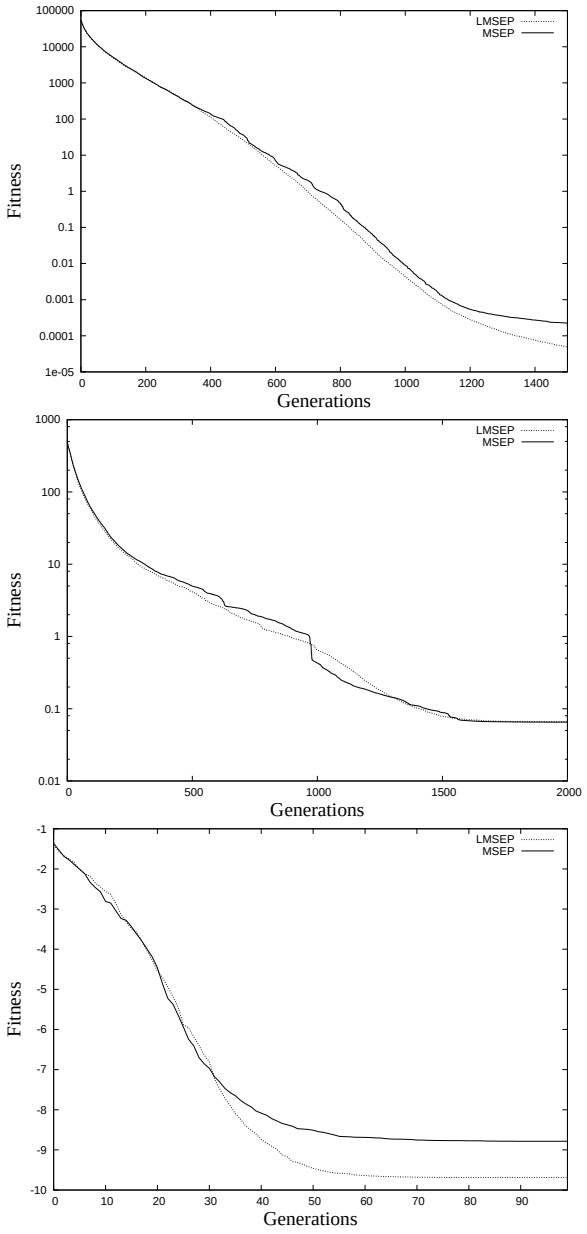

Fig. 1. Comparison between LMSEP and MSEP.

The stopping criterion is: to stop running at 1500 generations for functions f_1, f_6 f_{10} and f_{12}, 2000 generations for f_2 and f_{11}, 5000 generations for f_4 and f_9, 4000 generations for f_{15}. The rest will run 100 iterations. The lower-bound used for them is $\sigma_{\min} = 10^{-5}$ for all functions except f_4. Results for $f_1 - f_{15}$ are averaged over 50 independent runs, and for $f_{16} - f_{23}$ over 1000 independent trials.

The performance of EP using a mixed strategy mutation is compared with that of MSEP, SPMEP, LEP, FEP and CEP, whose results are presented in Table II and Table III. For simplicity, the algorithm proposed in this work (which is related to the local fitness landscape) is named LMSEP hereafter. MSEP [9] is an EP with a mixed mutation strategy in which different mutation operators are considered and adaptively employed according to a dynamic probabilistic distribution. CEP is an EP using a Gaussian mutation and FEP stands for EP using a Cauchy mutation which then is extended to LEP.

Table II lists the results of the mean best fitness generated in benchmark functions, from which it is observed that LMSEP produces a reasonable result at a similar level of precision which reveals that the mixed strategy performs at least as well as a pure strategy. However, it can be seen that, with the use of LEP or FEP, a considerably improved result for f_4, f_7, f_9 and f_{12} can be obtained. By exploring the mutation mechanism of LEP and FEP, the main reason can be considered as the relative higher flexibility of LEP and FEP than that of LMSEP. As the feature and mixed strategy is one-to-one correspondence in LMSEP, the update of mixed strategy would be stopped in certain generation once there is no improvement to the feature of local fitness landscape. As a result, the ability of producing large step jump is reduced at the later stage of the search [3]. It would be potentially favorable to assign a parameter, expressed as percentage, to let the process have a limit flexibility which is not in compliance with the one-to-one correspondence.

Fig. 1 provides a graphical comparison of average values of population found by two mixed strategies, LMSEP and MSEP, over 50 runs, where both algorithms employ two types of mutation operator, namely Gaussian mutation and Cauchy mutation. Two benchmark functions, f_1, f_{11} and f_{22}, are tested here. It is clear that LMSEP can search further on f_1 and f_{22} while MSEP can not find relative improved results. For f_{11}, LMSEP displays a faster convergence rate than MSEP. LMSEP quickly reaches approximately 1 in around 1000 generations. After that, MSEP exhibits a substantial descent and overtakes LMSEP. However finally, both algorithms reach approximatively same result around 1700 generations. Take all cases into account, the curves of process suggest a more efficient progress is introduced by LMSEP.

The standard deviation of LMSEP whose evaluation is given in Table III is also compared with those obtained using other approaches. According to the results, LMSEP exhibits similar performances on most functions except f_4, f_7, f_9 and f_{12}. This fact indicates that LMSEP has at least the same stability with a single pure strategy. However, LMSEP does not present a better performance on some function to which LEP and FEP are nicely applied. It suggests that the search of mixed strategy is sometimes not able to focus on the position where the global minimum is situated. It means the adjustment of the feature of local fitness landscape, the implementation of φ in this paper, remains to be carefully modified in future research so that a better result can be expected.

V. Conclusions

This paper has presented a new EP, LMSEP, which is characterized by the local fitness landscape using a mixture of different mutation strategies. The approach addresses the

TABLE III
COMPARISON OF STANDARD DEVIATION BETWEEN LMESP AND MSEP, LEP, FEP, CEP

	Generations	LMSEP	MSEP	LEP	FEP [3]	CEP [3]
		Std. dev	Std. dev	Std. dev	Std. dev	Std. dev
f_1	1500	7.971e-5	1.607e-4	6.043e-3	1.3e-4	5.9e-4
f_2	2000	9.37e-4	4.346e-1	3.136e-3	7.7e-4	1.7e-4
f_4	5000	1.09	1	2.059e-4	0.3	0.5
f_6	1500	0	126	0	0	1125.76
f_7	3000	1.854e-2	1.744e-2	2.486e-3	2.6e-3	6.4e-3
f_9	5000	13.18	13.8	15.127	4.6e-2	1.2e-2
f_{10}	1500	1.762e-3	2.485	2.813e-2	2.1e-3	2.8
f_{11}	2000	4.843e-2	11.198	1.114e-2	1.8e-2	9.2
f_{12}	1500	2.065	2.46e-2	5.7e-5	3.6e-6	2.4
f_{15}	4000	1.885e-4	1.090e-6	1.5e-4	1.8e-2	9.2
f_{16}	100	4.817e-9	3.417e-8	3.247e-3	4.9e-7	4.9e-7
f_{18}	100	4.431e-8	0	2.183e-1	0.11	0
f_{19}	100	7.22e-8	4.568e-7	1.19e-4	1.4e-5	1.4e-2
f_{21}	100	2.744	2.59	2.934	1.59	2.67
f_{22}	100	2.136	2.32	1.959	2.12	2.95
f_{23}	100	2.466	2.38	2.284	3.14	2.92

drawbacks of conventional EPs that employ a single mutation operator. In addition, a training procedure has been given to promote LMSEP in an efficient and intelligent way, by introducing a self-learning algorithm.

The performance of LMSEP is firstly trained on 7 functions and then tested on a suite of 16 benchmark functions, in comparison with previously studied EPs. The experimental evaluation indicates that the new approach has the ability to produce relatively more acceptable results. The tests regarding the standard deviation also demonstrate that LMSEP has a more robust performance.

Although the training procedure leads to a fast performance, it may occasionally miss certain regions that should be checked for. To address this issue, a fine adjustment of training procedure remains an active research. For instance, a backward jump procedure may be potentially employed. As a compatible satisfactory result can be obtained using FEP and LEP, a better implementation of the φ parameter may be useful. Additionally, more mutation operators can be taken into account, (e.g. Lévy mutation). Finally, introducing mixed strategies to other types of operator like crossover and selection also forms an interesting piece of future work.

REFERENCES

[1] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution*. New York, NY, USA: John Wiley & Sons, Inc., 1966.

[2] D. B. Fogel, *Evolutionary computation: toward a new philosophy of machine intelligence*. Piscataway, NJ, USA: IEEE Press, 1995.

[3] X. Yao, Y. Liu, , and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 82–102, 1999.

[4] C.-Y. Lee and X. Yao, "Evolutionary programming using mutations based on the lévy probability distribution," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 1, pp. 1–13, 2004.

[5] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[6] J. He and X. Yao, "A game-theoretic approach for designing mixed mutation strategies," in *ICNC'05 (LNCS 3612)*. Springer-Verlag, 2005, pp. 279–288.

[7] K. Chellapilla, "Combining mutation operators in evolutionary programming," *IEEE Transaction on Evolutionary Computation*, vol. 2, no. 3, pp. 91–96, 1998.

[8] H. Zhang and J. Lu, "Adaptive evolutionary programming based on reinforcement learning," *Information Science*, vol. 178, no. 4, pp. 971–984, 2008.

[9] H. Dong, J. He, H. Huang, and W. Hou, "Evolutionary programming using a mixed mutation strategy," *Information Sciences*, vol. 177, no. 1, pp. 312–327, 2007.

[10] L. Shen and J. He, "Evolutionary programming using a mixed strategy adapting to local fitness landscape," in *Proceedings of the 2009 UK Workshop on Computational Intelligence*. University of Nottingham, 2009, pp. 92–97.

[11] Y. Liu, "Operator adaptation in evolutionary programming," in *ISICA*, 2007, pp. 90–99.

[12] G. B. Fogel, G. B. Fogel, and K. Ohkura, "Multiple-vector self-adaptation in evolutionary algorithms," *BioSystems*, vol. 61, pp. 155–162, 2001.

[13] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill Science/Engineering/Math, March 1997.