A Tribal Ecosystem Inspired Algorithm (TEA) For Global Optimization

Ying Lin [a], Jing-Jing Li [b] (Corresponding Author), Jun Zhang [c] and Meng Wan [d]
[a] Department of Psychology, Sun Yat-sen University
[b] School of Computer Science, South China Normal University
[c] School of Advanced Computing, Sun Yat-sen University
[d] Center for Science and Technology Development, Ministry of Education
jingjing.li1124@gmail.com

ABSTRACT
Evolution mechanisms of different biological and social systems have inspired a variety of evolutionary computation (EC) algorithms. However, most existing EC algorithms simulate the evolution procedure at the individual-level. This paper proposes a new EC mechanism inspired by the evolution procedure at the tribe-level, namely tribal ecosystem inspired algorithm (TEA). In TEA, the basic evolution unit is not an individual that represents a solution point, but a tribe that covers a subarea in the search space. More specifically, a tribe represents the solution set locating in a particular subarea with a coding structure composed of three elements: *tribal chief*, *attribute diversity*, and *advancing history*. The tribal chief represents the locally best-so-far solution, the attribute diversity measures the range of the subarea, and the advancing history records the local search experience. This way, the new evolution unit provides extra knowledge about neighborhood profiles and search history. Using this knowledge, TEA introduces four evolution operators, *reforms*, *self-advance*, *synergistic combination*, and *augmentation*, to simulate the evolution mechanisms in a tribal ecosystem, which evolves the tribes from potentially promising subareas to the global optimum. The proposed TEA is validated on benchmark functions. Comparisons with three representative EC algorithms confirm its promising performance.

Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *heuristic methods.*

General Terms
Algorithms, Experimentation

Keywords
Evolutionary computation (EC), global optimization, tribe

1. INTRODUCTION
Evolutionary computation (EC) is a kind of probabilistic optimization algorithms inspired by the evolution mechanisms of complex systems in nature. Given an optimization problem, an EC algorithm is initialized with a population in which each individual represents a candidate solution. The fitness of individuals is evaluated in a way that the quality of represented solutions can be reflected proportionally. Evolution operators are designed to implement the principles of inheritance, variation, and selection in natural evolution. The EC algorithm performs the evolution operators for evolving the population towards the global optimum in an iterative manner.

The above evolution steps allow EC algorithms to solve an optimization problem without requiring any problem-specific knowledge other than a quantitative criterion for evaluating the quality of solutions. Such flexibility and simplicity encourage the applications of EC algorithms in a wide range of fields. However, despite the success of EC algorithms on solving many problems, they still suffer from the weakness of lengthy computational time and potentially premature convergence. Therefore, how to design algorithms that can find the global optimum in short time remains to be one of the most promising yet challenging research topics in the field of EC.

In the literature, the most notable and promising methods for addressing the above topic include adapting parameters to strike a dynamic balance between global exploration and local exploitation [1]-[3], designing novel evolution operators to achieve better search efficiency [4][5], applying niching methods to preserve the diversity of the population [6][7], hybridizing with local search techniques [8][9], and using multi-population strategies [10][11], etc. The underlying strategies of the above methods for improving the optimization accuracy and efficiency are based on introducing sophisticated operations or refinement in the entire optimization process. However, their algorithms remain *solution-based*. That is, the basic evolution unit in these EC algorithms is solutions - discrete points in the search space. Nevertheless, a solution can only provide the position and quality information of itself, but cannot provide information about the surrounding area (neighborhood). The lack of neighborhood information is the essential cause of bind and repetitive search, slowing down the search speed or even causing prematurity.

A tribe is a social group of people that unite under one chief and share similarities in physical and cultural attributes. In a tribal ecosystem comprising heterogeneous tribes, a tribe seeks development from both inner- and inter-tribe cooperation [12]. Inspired by the evolution mechanisms in tribal ecosystems, this paper proposes a new EC mechanism, namely tribal ecosystem inspired algorithm (TEA). In TEA, each tribe is mapped to the solution set in a particular subarea in the search space. Specifically, the best solution found in the subarea is analogous to the tribal chief and the range of the subarea is interpreted as the diversity of the tribe in attributes. The search experience of

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.

http://dx.doi.org/10.1145/2576768.2598253

exploiting the subarea is also attached to the tribe as a record of its self-advancing history. By doing so, TEA is differentiated from traditional solution-based EC algorithms as the basic evolution unit has been changed from discrete solution points to solution sets in different subareas in the search space.

Given an optimization problem, TEA initially partitions the search space into a set of zones with uniform size and initializes the tribal society by founding tribes on a subset of evenly distributed zones. After initialization, TEA performs four novel operators iteratively to simulate the evolution procedure of real tribes. In each generation, a *reforms* operator is first performed to reorganize the structure of the tribal society based on the similarities of existing tribes. An existing tribe may split into new tribes, merge into another tribe, or remain unchanged. Then a *self-advance* operator is carried out to simulate the process that a tribe seeks development through inner cooperation. As in tribal ecosystems, TEA gives tribes with fitter chiefs and more successful self-advancing history large opportunities to undergo self-advance. The inner-tribe cooperation in the self-advancing process brings in homogeneity, which is reflected by a decreasing diversity in the attributes of tribes. Thirdly, a *synergistic combination* operator is executed to emulate the inter-communication among tribes. New tribes are formed by combining attributes of existing tribes. Finally, TEA uses an *augmentation* operator to imitate the process that the tribal society extends to less populated areas in seek of more living resources. Such augmentation process results in the formation of new tribes in less explored subareas of the search space.

Using tribes as the basic evolution unit brings TEA a number of benefits. Firstly, the *reforms* operator compels the resulting tribes to represent solution sets in subareas different from each other. As a result, TEA avoids converging in a particular subarea and reduces the possibility of prematurity. Secondly, with the tribal structure, not only solutions, but also information of their surrounding neighborhoods, are inherited from generation to generation. The impact of the neighborhood information is twofold. First, it facilitates subsequent exploitation in their neighborhoods. Second, it guides TEA to efficiently identify promising subareas in the search space. The effectiveness and efficiency of TEA is validated on a diverse set of benchmark functions. Comparisons of TEA with a number of representative EC algorithms confirm the advantage of the tribal structure.

The remaining sections of this paper are organized as follows. Section 2 introduces details of the proposed TEA. Experimental results and comparisons are displayed in Section 3. Section 4 draws a conclusion to the whole paper.

2. THE PROPOSED TEA

Without loss of generality, we introduce the proposed TEA in the context of the following n-dimensional minimization problem:

$$\min_{\boldsymbol{l}\le\boldsymbol{x}\le\boldsymbol{u}} f(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{x}$ is a candidate solution in the search space $[\boldsymbol{l}, \boldsymbol{u}]$ and $f(\cdot)$ is the fitness function. In TEA, each variable in the optimization problem is considered as a personal attribute and a society $\boldsymbol{S}$ of N tribes is evolved. Each tribe T_i is characterized by a tribal chief $\boldsymbol{c}_i$, an attribute diversity $\boldsymbol{d}_i$, and a historical record h_i, i.e.,

$$T_i = <\boldsymbol{c}_i, \boldsymbol{d}_i, h_i>. \tag{2}$$

The chief $\boldsymbol{c}_i = (c_{i,1}, c_{i,2}, \ldots, c_{i,n})$ is the one that achieves the best fitness in T_i. $c_{i,j}$ denotes the j-th attribute of $\boldsymbol{c}_i$. The diversity $\boldsymbol{d}_i = (d_{i,1}, d_{i,2}, \ldots, d_{i,n})$ measures the diversity of T_i on all the n attributes. The smaller $d_{i,j}$ is, the more homogeneity T_i has on the j-th attribute. With $\boldsymbol{c}_i$ as the centroid and $\boldsymbol{d}_i$ as the radius, the coverage of T_i can be described as $[\boldsymbol{c}_i - \boldsymbol{d}_i, \boldsymbol{c}_i + \boldsymbol{d}_i]$. The historical record h_i records the self-advancing history of T_i. A larger h_i value implies a greater success of T_i in the previous self-advancing process. Treating tribes coded by the above structure as the basic evolution unit, the evolution procedure of TEA comprises the following five steps.

2.1 Initialization

In the initialization phase, TEA chooses a parameter m and divides each dimension of the search space into m even segments. The search space is thus partitioned into (m^n) zones with uniform size. M zones are selected from the search space according to a pre-generated *orthogonal array* [13], where M is the total of level combinations in the array. The tribal society $\boldsymbol{S}$ is initialized with the chief of each tribe randomly generated in a selected zone. Suppose that '⊗' denotes a inner multiplication operator between two vectors with equal dimensionality. Then the initialization of $\boldsymbol{c}_i$ can be formulated as

$$\boldsymbol{c}_i = \boldsymbol{l} + (\boldsymbol{com}_i + \boldsymbol{r}_i - 1) \otimes (\boldsymbol{u} - \boldsymbol{l}) / m, \tag{3}$$

where $\boldsymbol{r}_i \in (0,1)^n$ is an n-dimensional normalized random vector and $\boldsymbol{com}_i \in [1,m]^n$ is the i-th level combination in the orthogonal array. The attribute diversity $\boldsymbol{d}_i$ and the historical record h_i are initialized as their initial values $\boldsymbol{d}_0$ and h_0, respectively. For the sake of simplicity in the implementation, $\boldsymbol{d}_0$ and h_0 are calculated here by

$$\boldsymbol{d}_0 = (\boldsymbol{u} - \boldsymbol{l})/m, \tag{4}$$

$$h_0 = \exp(\varepsilon_{\min}), \tag{5}$$

where $\varepsilon_{\min} \in (0, 1)$ is a predefined constant.

The orthogonal property of orthogonal arrays guarantees that the M selected zones are uniformly distributed in the search space [13]. The tribal society initialized by the above method is thus able to scan the search space evenly, which helps TEA to gain a general image of the overall fitness distribution.

2.2 Reforms

The tribes are reformed by clustering existing tribes on both scales of fitness and attributes. A clustering technique termed *polythetic division* is applied for doing so. This technique is used because it can achieve satisfying clustering performance without the need of setting extra parameters or using iterative optimization procedure. Due to length limit, please refer to [14] for details of polythetic division. Detailed steps of the reform process are introduced afterwards.

Successful application of polythetic division requires a distance metric and a criterion for judging whether a group can be further divided. In the proposed TEA, the distance $\delta_{i,k}$ between two tribes T_i and T_k ($i, k = 1, 2, \ldots, N, i \neq k$) is defined as follows:

$$\delta_{i,k} = \begin{cases} \left| f(\boldsymbol{c}_i) - f(\boldsymbol{c}_k) \right|, & \text{on the scale of fitness} \\ \sqrt{(\boldsymbol{c}_i - \boldsymbol{c}_k)^{\mathrm{T}} (\boldsymbol{c}_i - \boldsymbol{c}_k)}, & \text{on the scale of attributes} \end{cases}. \tag{6}$$

With respect to the dividable criterion, it is defined as the maximum depth of the corresponding classification tree. With the above definition, the reforming process of TEA is given below:

Step 1) Apply the polythetic division technique to cluster the N tribes in $\boldsymbol{S}$ until the depth of the classification tree reaches the predefined upper bound D_f. Denote the resulting set of groups as $\boldsymbol{G}' = \{G_1', G_2', ..., G_{N'}'\}$, where N' is the number of groups in $\boldsymbol{G}'$ and $N \le 2^{D_f}$.

Step 2) Apply the polythetic division technique to further cluster the tribes in each group $G_i' \in \boldsymbol{G}'$ on the scale of attributes. This clustering process continues until the depth of the corresponding classification tree reaches the predefined upper bound D_a. Denote the set of obtained groups as $\boldsymbol{G} = \{G_1, G_2, \ldots, G_N\}$, where N is the number of groups in $\boldsymbol{G}$ and $N \le 2^{D_f+D_a}$.

Step 3) A new tribe $T_i' = <\boldsymbol{c}_i', \boldsymbol{d}_i', h_i'>$ is defined based on each group $G_i \in \boldsymbol{G}$. Suppose that $T_{\text{best}}^{(i)} = <\boldsymbol{c}_{\text{best}}^{(i)}, \boldsymbol{d}_{\text{best}}^{(i)}, h_{\text{best}}^{(i)}> \in G_i$ is the tribe whose chief has the best fitness among all the tribes in G_i. Then T_i' is defined as

$$\boldsymbol{c}_i' = \boldsymbol{c}_{\text{best}}^{(i)}, \tag{7}$$

$$d_{i,j}' = \frac{1}{2}\left[\max_{T_k \in G_i}(c_{k,j} + d_{k,j}) - \min_{T_k \in G_i}(c_{k,j} - d_{k,j})\right], j = 1, 2, ..., n \tag{8}$$

$$h_i' = h_{\text{best}}^{(i)}. \tag{9}$$

The N tribes derived from the groups in $\boldsymbol{G}$ compose a new generation of $\boldsymbol{S}$.

Figure 1 exemplifies the above reforming process on a one-dimensional search space with $\boldsymbol{S} = \{T_1, T_2, \ldots, T_{10}\}$, $D_f = 2$, and $D_a = 1$. It can be observed that with $D_f = 2$, the ten tribes in $\boldsymbol{S}$ are divided into four groups on the scale of fitness, i.e., $G_1' = \{T_1, T_3, T_7, T_{10}\}$, $G_2' = \{T_4, T_6, T_8, T_9\}$, $G_3' = \{T_2\}$, and $G_4' = \{T_5\}$. Then the reforming process proceeds into the scale of attributes, as indicated by the arc arrows in Figure 1. Tribes in the above four groups are further clustered based on the attributes of their chiefs. With D_a=1, G_1' and G_2' are further divided into four groups as $G_1 = \{T_8, T_9\}$, $G_2 = \{T_4, T_6\}$, $G_3 = \{T_1, T_3\}$, and $G_4 = \{T_7, T_{10}\}$. G_3' and G_4' are not further divided as they have only one tribe. Consequently, with each of the six groups G_1, G_2, G_3, G_4, G_3', and G_4' defining a tribe, the reforming process reorganizes a ten-tribe society into a society of six tribes.

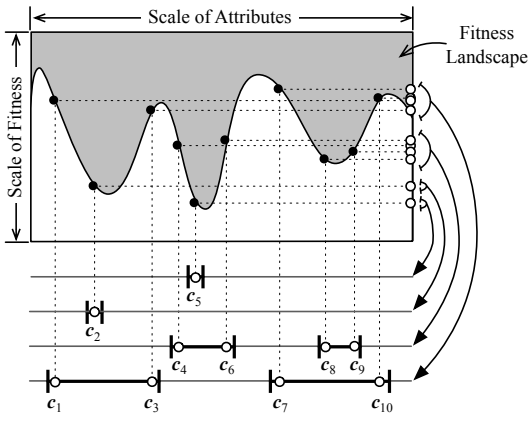

Figure 1. An example of the reforming process in TEA.

2.3 Self-advance

The probability for a tribe to self-advance depends on two factors: the leading power of its chief and its self-advancing history. A chief with better fitness is supposed to have the highest leading power, while a larger value of the historical record suggests a more successful self-advancing history. Suppose that in comparison with T_i, the society $\boldsymbol{S}$ contains F_i tribes that have less powerful chiefs and H_i tribes that have less successful self-advancing history. The self-advancing probability of T_i is calculated as follows:

$$p_{\text{adv}}(i) = \eta - (\eta - 1)\left[N - \sum_{1 \le k \le N}^{k \ne i} I_{F,H}(i,k)\right], \tag{10}$$

where $\eta \le 1$ is a coefficient and $I_{F,H}(i,k)$ is a function indicating whether or not T_i surpasses T_k by satisfying one of the following two criteria: (i) $F_i + H_i > F_k + H_k$ or (ii) $F_i + H_i = F_k + H_k$ and $F_i > F_k$, i.e.,

$$I_{F,H}(i,k) = \begin{cases} 1, & \text{if (i) or (ii) satified} \\ 0, & \text{otherwise} \end{cases}, \tag{11}$$

From (10) and (11), it can be known that tribes with more powerful chiefs and more successful self-advancing history are given higher self-advancing probabilities, which is consistent with the situations in tribal ecosystems.

To avoid the trouble of tuning η, the value of η is set as:

$$\eta = \exp\left(\frac{E}{E_{\max}}\right) \tag{12}$$

where $E > 0$ is the number of fitness evaluations (FEs) and $E_{\max}$ is the predefined upper limit of E. By doing so, η increases from one to e as the algorithm proceeds. The growing η, on the one hand, facilitates TEA to explore the search space in the early phase because a small η allows more tribes to do self-advancement. On the other hand, as the η value increases, TEA focuses more on exploiting several most promising tribes and thus accelerates the convergence speed.

Based on the self-advancing probabilities calculated as above, a subset of tribes is randomly selected from the society. The self-advancing process of each selected tribe T_i iterates a direction prediction phase and a directional advance phase. As shown in Figure 2, detailed implementation of the above two phases are described as follows:

1) Phase of Direction Prediction: In this phase, an advancing direction $\boldsymbol{adv}_i = (adv_{i,1}, adv_{i,2}, \ldots, adv_{i,n})$ is predicted through the cooperation of the tribal population in T_i. In detail, a set of M trial points is sampled from the population according to a pre-generated orthogonal array with n factors and three levels per factor. The three levels correspond to three actions on a value, which are increment, decrement, and unchanged, respectively. Suppose that the k-th level combination in the orthogonal array is denoted as $\boldsymbol{com}_k = (com_{k,1}, com_{k,2}, \ldots, com_{k,n})$, where $com_{k,j} \in [1,3]$ is the level of the j-th factor in $\boldsymbol{com}_k$, $k = 1, 2, \ldots, M$, and $M = 3^{\lceil \log_3(3n-2) \rceil}$. The k-th trial point $\boldsymbol{v}_k$ is generated by $\boldsymbol{c}_i + \boldsymbol{d}_i \otimes (\boldsymbol{com}_k - 2)$. Based on the fitness of the M trial points, the advancing direction $\boldsymbol{adv}_i$ is predicted as

$$adv_{i,j} = -2 + \arg\min_{1 \le lev \le 3} \sum_{\boldsymbol{v}_k \in V_{j,lev}} f(\boldsymbol{v}_k), j = 1,2,\ldots,n, \tag{13}$$

where $V_{j,lev} = \{\boldsymbol{v}_k \mid com_{k,j} = lev,\ k = 1, 2, \ldots, M,\ lev \in [1, 3]\}$ is a subset of the M trial points. To validate whether the above prediction successfully predicts a beneficial direction, a trial point $\boldsymbol{z}_0$ is generated as

$$\boldsymbol{z}_0 = \boldsymbol{c}_i + \boldsymbol{d}_i \otimes \boldsymbol{adv}_i \,. \tag{14}$$

If $f(\boldsymbol{z}_0)$ is better than $f(\boldsymbol{c}_i)$, the prediction is successful. The self-advancing process enters into the phase of directional advance for seeking further improvement. Conversely, the prediction fails when $f(\boldsymbol{z}_0)$ is no better than $f(\boldsymbol{c}_i)$. In this case, $\boldsymbol{d}_i$ is shrunk by multiplying a parameter ρ_p. Suppose that α failures have occurred consecutively for predicting a beneficial direction, $\alpha \geq 1$. ρ_p is calculated as $\max(0.1, 0.9^{\alpha})$. The self-advancing process keeps trying to predict a beneficial direction until the total φ_p of unsuccessful prediction exceeds a predefined number Φ_p.

2) Phase of Directional Advance: The phase of directional advance is performed only when a beneficial direction is successfully predicted. In this phase, T_i continues to advance along $\boldsymbol{adv}_i$ step by step. For each advancing step k ($k \geq 1$), a trail point $\boldsymbol{z}_k$ is generated as

$$\boldsymbol{z}_k = \boldsymbol{z}_{k-1} + \boldsymbol{d}_i \otimes \boldsymbol{adv}_i \,. \tag{15}$$

The step succeeds if $f(\boldsymbol{z}_k)$ is better than $f(\boldsymbol{z}_{k-1})$. Otherwise, the step fails and $\boldsymbol{z}_k$ is replaced by $\boldsymbol{z}_{k-1}$. After each advancing step, $\boldsymbol{d}_i$ is adjusted as

$$\boldsymbol{d}_i = \begin{cases} \rho_s \boldsymbol{d}_i, & \text{if } f(\boldsymbol{z}_{k-1}) \leq f(\boldsymbol{z}_k) \\ \boldsymbol{d}_i / \rho_s, & \text{otherwise} \end{cases} , \tag{16}$$

where ρ_s is calculated as $\max(0.1, 0.9^{\beta})$ and β is the number of consecutive successful (or unsuccessful) steps. The directional advance continues until the total φ_s of unsuccessful steps exceeds a predefined number Φ_s. Suppose that a total of λ steps have been performed by then. $\boldsymbol{c}_i$ is replaced by $\boldsymbol{z}_{\lambda}$ and h_i is updated as

$$h_i = \exp\left[\left(f_0 - f(\boldsymbol{c}_i)\right) / \left|f_0\right|\right] \tag{17}$$

where f_0 denotes the fitness of $\boldsymbol{c}_i$ at the very beginning of the self-advancing process.

The remaining problem of the above self-advancing process is how to set Φ_p and Φ_s. Since the optimal settings of Φ_p and Φ_s may vary in different problems or even different phases of the algorithm, it is more desired that the parameters can be tuned automatically after reasonable initial values are given. In this paper, we design an adaption strategy that adjusts Φ_p and Φ_s based on the improvement obtained. More specifically, the adjustment of Φ_p is made according the relative improvement achieved in one iteration of the self-advancing process, whereas the adjustment of Φ_s is made based on the relative improvement obtained from one step in the phase of directional advance. Detailed adaption rules are listed as follows:

Rule 1) Φ_p (or Φ_s) is increased by one if the relative improvement obtained in the current iteration (or step) exceeds the upper bound ξ_{max}, which is defined as

$$\xi_{max} = 10^{1+\lfloor \log_{10}(\Delta f) \rfloor} , \tag{18}$$

where Δf denotes the relative improvement obtained from the previous iteration (or step).

Rule 2) Φ_p (or Φ_s) is reduced by one if the relative improvement obtained in the current iteration (or step) drops below the lower bound ξ_{min} but remains non-zero. ξ_{min} is defined as

$$\xi_{min} = \min(\varepsilon_{min}, 10^{\lfloor \log_{10}(\Delta f) \rfloor}) \,. \tag{19}$$

Rule 3) Φ_p (or Φ_s) is reduced by one if no improvement has been achieved for two consecutive iterations (or steps).

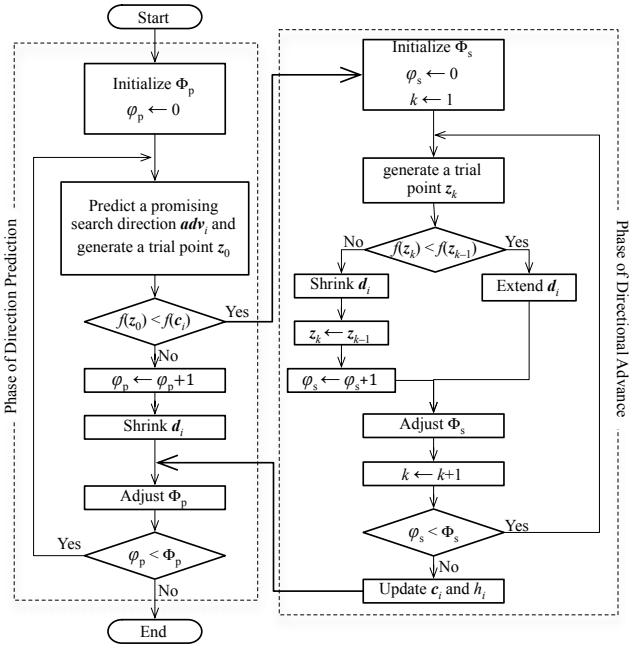

Figure 2. Flowchart of the self-advancing process of a tribe T_i.

2.4 Synergistic Combination

Among the N tribes in the society $\boldsymbol{S}$, the tribe whose chief has the best fitness is termed the *major tribe*. Without loss of generality, suppose that T_1 is the major tribe. Then synergy between T_1 and each of the other (N–1) tribes occurs and generates (N–1) new tribes. All the new tribes coexist with the original N tribes until the reforms operator reorganizes $\boldsymbol{S}$ in the next generation. Denote the new tribe generated by the synergy between T_1 and T_k as T_k^{syn}. The chief $\boldsymbol{c}_k^{syn}$ of T_k^{syn} combines the attributes of $\boldsymbol{c}_1$ and $\boldsymbol{c}_k$ as

$$c_k^{syn} = \begin{cases} c_{k,j}, & \text{if } r_{k,j} \leq (N - R_k)/2N \\ c_{1,j}, & \text{otherwise} \end{cases} , \tag{20}$$

where $r_{k,j} \in (0, 1)$ is a normalized random number and $R_k \in [1, N]$ is the rank of T_k in order of ascending fitness values. The attribute diversity and the historical record of T_k^{syn} are set as $\boldsymbol{d}_0$ and h_0, respectively.

From (20), it can be deduced that c_k^{syn} is more likely to be $c_{1,j}$ when R_i increases. Thus, the chief of a new tribe inclines to learn from the chief of the major tribe if the other tribe in synergy has a poor chief. The synergism process thus increases the chance for the new chiefs to achieve good fitness.

2.5 Argumentation

TEA keeps a census $\boldsymbol{U} = [u_{j,k}]_{n \times m}$ of the tribal society since the beginning of the algorithm. Each element $u_{j,k}$ in the census $\boldsymbol{U}$ records the number of tribal chiefs whose attribute on the j-th dimension falls into the range of

$$seg(j,k)=\left[\, l_j+\frac{(k-1)(u_j-l_j)}{m}, l_j+\frac{k(u_j-l_j)}{m}\,\right]. \qquad (21)$$

The probability to augment into $seg(j, k)$ is calculated based on the reverse of $u_{j,k}$ as

$$p_{\text{aug}}(j,k)=\frac{1}{1+u_{j,k}}\ . \qquad (22)$$

By doing so, augmentation is biased towards areas with smaller population.

Based on the probabilities in (22), the augmentation process generates $2^{D_f+D_a}$ new tribes, all of which will enter the society. In detail, the chief $\boldsymbol{c}_k^{\text{aug}}$ of the i-th new tribe T_k^{aug} is generated as

$$c_{k,j}^{\text{aug}}=l_j+(K-1+r_{i,j})(u_j-l_j)/m\ , j = 1, 2, \ldots, n, \qquad (23)$$

where $r_{i,j} \in (0, 1)$ is a normalized random number and $K \in [1, m]$ denotes the segment picked out by the *roulette wheel selection* based on the probability distribution in (22). The attribute diversity and the historical record of T_k^{aug} are set as $\boldsymbol{d}_0$ and h_0, respectively.

TEA completes an evolution cycle after performing the augmentation operator. Since both the synergistic combination and augmentation operators generate new tribes, the number of tribes in the society rises, causing the N value to update after each evolution cycle.

3. EXPERIMENTS AND DISCUSSIONS

In this section, TEA is applied to minimize a set of seventeen benchmark functions with thirty dimensions (n = 30) [15][16]. Although the global minima of these functions are all zero, their properties are significantly different. The first four functions f_1 to f_4 are unimodal problems. Conducting experiments on these unimodal functions helps us to analyze the local search ability of TEA. The next eight functions f_5 to f_{12} are multimodal functions, each of which has a unique global optimum and multiple local optima. Experiments on these multimodal functions provide a systematic view into the ability of TEA in performing global search and avoiding prematurity. The last five functions f_{13} to f_{17} are shifted & rotated multimodal functions proposed by Suganthan *et al.* [16]. We test TEA on these functions for evaluating its performance on complex problems. Detailed definitions of the above test functions are given in Table 1. As can be seen, each function is attached with an accuracy level, which is the maximum error allowed for an acceptable solution.

Table 1 Benchmark Functions

Function	Search Domain	Accuracy Level
$f_1(\boldsymbol{x})=\sum_{i=1}^{n} x_i^2$	[–100, 100]n	10-10
$f_2(\boldsymbol{x})=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	[–10, 10]n	10-10
$f_3(\boldsymbol{x})=\sum_{i=1}^{n}\left(\sum_{i=1}^{i} x_j\right)^2$, $\boldsymbol{y}=\boldsymbol{x}-\boldsymbol{o}$	[–100, 100]n	10-10
$f_4(\boldsymbol{x})=\sum_{i=1}^{n}\lfloor x_i+0.5\rfloor^2$	[–100, 100]n	10-10
$f_5(\boldsymbol{x})=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i)^2+(x_i-1)^2]$	[–30, 30]n	100
$f_6(\boldsymbol{x})=-\sum_{i=1}^{n} x_i\sin(\sqrt{\lvert x_i\rvert})+418.9829n$	[–500, 500]n	10-10
$f_7(\boldsymbol{x})=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$	[–5.12, 5.12]n	10-10
$f_8(\boldsymbol{x})=-20\exp\left[-0.2\sqrt{\sum_{i=1}^{n} x_i^2/n}\right]-\exp\left[\sum_{i=1}^{n}\cos(2\pi x_i)/n\right]+1$	[–32, 32]n	10-10
$f_9(\boldsymbol{x})=\sum_{i=1}^{n} x_i^2/4000-\prod_{i=1}^{n}\cos(x_i/\sqrt{i})+1$	[–600, 600]n	10-10
$f_{10}(\boldsymbol{x})=\frac{\pi}{20}\{10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n} u(x_i,10,100,4)$, where $\boldsymbol{y}=1+(\boldsymbol{x}+1)/4$	[–50, 50]n	10-10
$f_{11}(\boldsymbol{x})=1/10\{10\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)[1+\sin^2(3\pi x_{i+1})]+(x_n-1)^2[1+\sin^2(2\pi x_n)]\}+\sum_{i=1}^{n} u(x_i,5,100,4)$	[–50, 50]n	10-10
$f_{12}(\boldsymbol{x})=\sum_{i=1}^{n}\left[\sum_{j=1}^{n}(a_{i,j}\sin\alpha_j+b_{i,j}\cos\alpha_j)-\sum_{j=1}^{n}(a_{i,j}\sin x_j+b_{i,j}\cos x_j)\right]^2$	[–π, π]n	100
$f_{13}(\boldsymbol{x})=f_8(\boldsymbol{z})$, $\boldsymbol{z}=(\boldsymbol{x}-\boldsymbol{o})\cdot\boldsymbol{M}$	[–32, 32]n	21
$f_{14}(\boldsymbol{x})=f_7(\boldsymbol{z})$, z = $(\boldsymbol{x}-\boldsymbol{o})\cdot\boldsymbol{M}$	[–5, 5]n	100
$f_{15}(\boldsymbol{x})=\sum_{k=0}^{20}\left\{\sum_{i=1}^{n} 2^{-k}\cos\left[2\pi 3^k(z_i+0.5)\right]\right\}-n\sum_{k=0}^{20} 2^{-k}\cos(3^k\pi)$, where $\boldsymbol{z}=(\boldsymbol{x}-\boldsymbol{o})\cdot\boldsymbol{M}$	[–0.5, 0.5]n	20
$f_{16}=\sum_{i=1}^{n-1} f_9\left(f_5(z_i,z_{i+1})\right)+f_9\left(f_5(z_n,z_1)\right)$, where $\boldsymbol{z}=\boldsymbol{x}-\boldsymbol{o}+1$	[–5, 5]n	10
$f_{17}(\boldsymbol{x})=\sum_{i=1}^{n-1} F(z_i,z_{i+1})+F(z_n,z_1)$, where z = (x – o)·M	[–100, 100]n	13

* In f10 and f11, the function u is defined as $u(x,a,k,m)=\begin{cases} k(x-a)^m, & \text{if } x>a \\ k(-x-a)^m, & \text{if } x<-a \\ 0, & \text{otherwise}\end{cases}$.

* In f12, $a_{i,j}$ and $b_{i,j}$ are random integers in the range of [–100, 100] and α_j is a random number in the range of [–π, π], i=1,2,…,n, j=1,2,…,n.
* In f13 to f17 $\boldsymbol{o}$ denotes a random vector in the search domain and $\boldsymbol{M}$ denotes an n by n orthogonal matrix.

* In f_{17}, the function F is defined as $F(x,y)=0.5+\frac{\sin^2(\sqrt{x^2+y^2})-0.5}{\left[1+0.001(x^2+y^2)\right]^2}$

To further validate the effectiveness and efficiency of TEA, its results are compared with three representative EC algorithms: evolutionary parallel local search (EPLS) [6], comprehensive learning particle swarm optimization (CLPSO) [4], and an adaptive differential evolution algorithm with an optional external archive (JADE) [3]. For the sake of fairness in the comparisons, all the four algorithms are programmed and complied with Visual C++ 6.0 in Windows XP (Service Pack 2). EPLS, CLPSO, and JADE use exactly the same parameter settings as their original papers, while the parameters of TEA are set as $m = 9$, $D_f = 2$, $D_a = 3$, $\varepsilon_{\min} = 10^{-3}$, and $\Phi_p = \Phi_s = 2$. Using 300,000 FEs as the termination criterion ($E_{\max} = 300{,}000$), the four algorithms report their average results obtained from fifty independent trials for comparison.

3.1 Unimodal Functions

Table 2 tabulates the results on the unimodal functions f_1 to f_4. As can be observed, TEA achieves the best solution accuracy on f_1 and f_2, followed by JADE, CLPSO, and EPLS. JADE obtains the best result on f_3, but the Mann-Whitney test indicates that TEA is still significantly better than EPLS and CLPSO. Each algorithm except for EPLS is able to find the global minimum zero on the non-continuous step function f_4 with a 100% successful rate. In general, TEA obtains the best solution accuracy on the four unimodal functions.

Table 3 tabulates the average FEs number for the algorithms to find the first acceptable solution. The successful rates, that is, the percentage of successful trials in which acceptable solutions are found, are also reported. From the table, it can be observed that on f_1, f_2, and f_4, TEA, JADE, and CLPSO all obtain 100% successful rates on the accuracy level at 10^{-10}. However, the average FEs number needed by TEA is significantly less than those of the other two algorithms. Take f_1 as an example: the average FEs number needed by TEA is 5596, which is less than 1/6 of the smallest average FEs number among the other three algorithms (JADE: 35313). The large gap in the average FEs number reveals the advantage of TEA in search speed. On f_3, TEA and JADE are the only two algorithms that find acceptable solutions on the accuracy level at 10^{-10}. JADE leads TEA in the successful rate by 4% and the average FEs number needed by JADE is smaller than that of TEA.

Table 2. Comparison of Solution Accuracy on Unimodal Functions[1]

Func	Alg	Best	Mean	Std.	Significance[2]
f_1	EPLS	5.13×10^{-28}	1.64×10^{-8}	7.37×10^{-8}	+1
	CLPSO	5.15×10^{-24}	2.36×10^{-23}	1.32×10^{-23}	+1
	JADE	2.26×10^{-149}	3.03×10^{-83}	2.14×10^{-82}	+1
	TEA	**0**	**0**	**0**	-
f_2	EPLS	8.01×10^{-14}	6.59×10^{-4}	3.58×10^{-3}	+1
	CLPSO	7.27×10^{-15}	1.41×10^{-14}	4.77×10^{-15}	+1
	JADE	2.15×10^{-76}	1.16×10^{-30}	8.19×10^{-30}	+1
	TEA	**0**	**9.08×10^{-141}**	**6.42×10^{-140}**	-
f_3	EPLS	2.94×10^{-8}	0.16596	0.51147	+1
	CLPSO	68.817	170.75	48.718	+1
	JADE	**7.91×10^{-42}**	**1.44×10^{-37}**	**5.05×10^{-37}**	−1
	TEA	5.03×10^{-41}	6.43×10^{-10}	3.18×10^{-9}	-
f_4	EPLS	0	0.92	1.9984	+1
	CLPSO	**0**	**0**	**0**	0
	JADE	**0**	**0**	**0**	0
	TEA	**0**	**0**	**0**	-

[1] The results marked in **boldface** are the best among the four algorithms.
[2] '+1', '−1', and '0' in the significance column indicate that compared to the result of TEA, the annotated result is significantly worse, significantly better, or insignificantly different, respectively.

Table 3. Comparison of Search Speed & Convergence Reliability on Unimodal Functions[1]

Func	Average FEs Number				Successful Rate (%)			
	EPLS	CLPSO	JADE	TEA	EPLS	CLPSO	JADE	TEA
f_1	183274	173335	35313	**5596**	76	**100**	**100**	**100**
f_2	196110	235871	63435	**54710**	12	**100**	**100**	**100**
f_3	-	-	**108117**	208264	0	0	**100**	96
f_4	138838	49624	9379	**2211**	64	**100**	**100**	**100**

[1] The results marked in **boldface** are the best among the four algorithms.

Figure 3 depicts the convergence curves of the four algorithms by plotting their median results obtained in the fifty trials along the FEs number. As can be observed, on f_1, f_2, and f_4, the convergence curves of TEA are much steeper than those of the other three algorithms. The advantage of TEA in search speed is thus confirmed on these three functions. With respect to f_3, the convergence curve of JADE exhibits the fastest descending rate. The convergence curve of TEA on f_3 is close to that of JADE and is much steeper than those of EPLS and CLPSO.

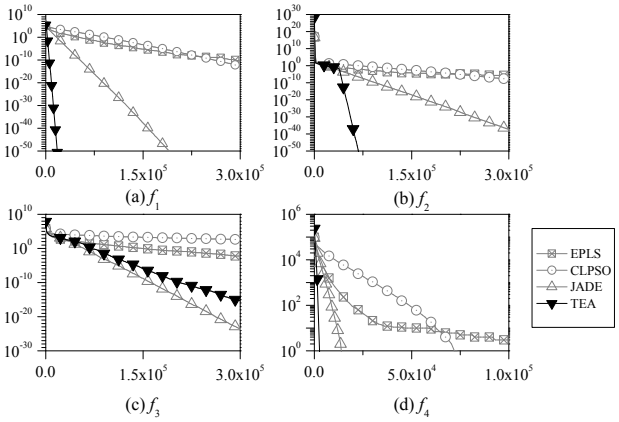

Figure 3. Convergence curves of EPLS, CLPSO, JADE, and TEA on unimodal functions f_1 to f_4.

The above results prove that TEA is capable of finding high-quality solutions to unimodal problems at fast speed. Compared to the other three algorithms, TEA possesses general advantage in both terms of solution accuracy and search speed. This is because the adaptive self-advancing processes of tribes endow TEA with powerful local search ability. Besides, since tribes with fitter chiefs and better self-advancing history are given higher self-advancing probabilities, TEA can bias exploitation towards promising areas where better solutions are more likely to be found. By doing so, TEA accelerates the convergence towards the global optimum.

3.2 Multimodal Functions

According to the results reported in Table 4, among the eight multimodal functions f_5 to f_{12}, the proposed TEA obtains the best average results on f_6, f_7, f_8, f_9, f_{10}, f_{11}, and f_{12}. Specifically, TEA completely avoids premature convergence on f_6 to f_{11}. The other algorithm that can do so is CLPSO. However, as shown by the average results on f_8 to f_{11}, the solution accuracy of CLPSO is significantly worse because CLPSO has to sacrifice the convergence speed for performing global exploration [4].

Table 5 reports the average FEs numbers and the successful rates for the four algorithms to find acceptable solutions on f_5 to f_{12}. As can be observed from the table, TEA always achieves the highest successful rate with the minimum or second minimum number of FEs. Specifically, compared to the other algorithms that achieve

100% successful rates on f_7 to f_{11}, the FEs number needed by TEA is much smaller.

Table 4. Comparison of Solution Accuracy on Multimodal Functions[1]

Func	Alg	Best	Mean	Std.	Significance[2]
f_5	EPLS	0.3605	71.27	45.60	+1
	CLPSO	0.09826	18.07	21.00	−1
	JADE	**0**	**0.07973**	**0.5638**	−1
	TEA	10.46	25.33	14.47	-
f_6	EPLS	355.3	934.1	341.12	+1
	CLPSO	**3.82×10^{-4}**	**3.82×10^{-4}**	3.60×10^{-13}	0
	JADE	**3.82×10^{-4}**	11.84	35.892	+1
	TEA	**3.82×10^{-4}**	**3.82×10^{-4}**	**3.06×10^{-13}**	-
f_7	EPLS	8.955	16.62	5.018	+1
	CLPSO	**0**	**0**	**0**	0
	JADE	**0**	**0**	**0**	0
	TEA	**0**	**0**	**0**	-
f_8	EPLS	5.46×10^{-11}	1.975	0.7895	+1
	CLPSO	7.55×10^{-15}	8.12×10^{-15}	2.42×10^{-15}	+1
	JADE	**4.00×10^{-15}**	**4.00×10^{-15}**	**0**	0
	TEA	**4.00×10^{-15}**	**4.00×10^{-15}**	**0**	-
f_9	EPLS	**0**	0.01632	0.02856	+1
	CLPSO	**0**	8.05×10^{-15}	2.99×10^{-14}	+1
	JADE	**0**	1.45×10^{-5}	1.05×10^{-3}	+1
	TEA	**0**	**0**	**0**	-
f_{10}	EPLS	2.25×10^{-27}	0.02695	0.08587	+1
	CLPSO	3.30×10^{-25}	1.94×10^{-24}	1.16×10^{-24}	+1
	JADE	**1.57×10^{-32}**	**1.57×10^{-32}**	**8.29×10^{-48}**	0
	TEA	**1.57×10^{-32}**	**1.57×10^{-32}**	**8.29×10^{-48}**	-
f_{11}	EPLS	1.26×10^{-20}	0.3626	0.7067	+1
	CLPSO	1.27×10^{-23}	8.47×10^{-23}	5.41×10^{-23}	+1
	JADE	**1.35×10^{-31}**	**1.35×10^{-31}**	**1.77×10^{-46}**	0
	TEA	**1.35×10^{-31}**	**1.35×10^{-31}**	**1.77×10^{-46}**	-
f_{12}	EPLS	565.6	10682	7665.6	+1
	CLPSO	2867.3	7860.1	2666.4	+1
	JADE	**1.84×10^{-12}**	9655.5	5822.3	+1
	TEA	0.1016	**4749.1**	**4105.8**	-

[1] The results marked in **boldface** are the best among the four algorithms.
[2] '+1', '−1', and '0' in the significance column indicate that compared to the result of TEA, the annotated result is significantly worse, significantly better, or insignificantly different, respectively.

Table 5. Comparison of Search Speed & Convergence Reliability on Multimodal Functions[1]

Func	Average FEs Number				Successful Rate (%)			
	EPLS	CLPSO	JADE	TEA	EPLS	CLPSO	JADE	TEA
f_5	79265	89115	11088	42471	88	**100**	**100**	**100**
f_6	-	149170	**114205**	163962	0	**100**	90	**100**
f_7	-	210691	158198	**112507**	0	**100**	**100**	**100**
f_8	239755	107912	56732	**10293**	2	**100**	**100**	**100**
f_9	188330	225950	38760	**12005**	38	**100**	98	**100**
f_{10}	174678	160845	33025	**9343**	68	**100**	**100**	**100**
f_{11}	206296	177754	42602	**23496**	46	**100**	**100**	**100**
f_{12}	-	-	**109457**	116074	0	0	4	**6**

[1] The results marked in **boldface** are the best among the four algorithms.

The above results demonstrate the effectiveness and efficiency of TEA in solving multimodal problems. Figure 4 depicts the convergence curves of the four algorithms on f_5 to f_{12} to further illustrate the search behavior of TEA on multimodal problems. From the figure, it is noticed that the search speed of TEA can be relatively slow in the early stage of the optimization procedure (as shown in Figure 4 (b), (c), and (g)). This is because TEA is trying to identify the basin of the global minimum from the set of areas defined by the tribes. Once the basin is found, the self-advancing process of the corresponding tribe will quickly bring the search procedure to the precise location of the global optimum.

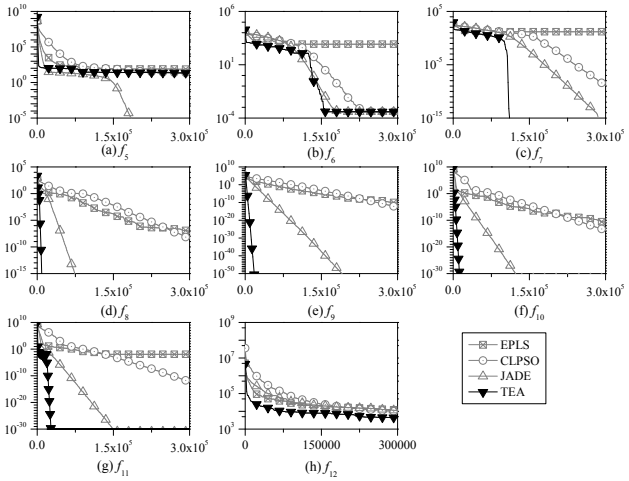

Figure 4. Convergence curves of EPLS, CLPSO, JADE, and TEA on multimodal functions f5 to f12.

3.3 Shifted & Rotated Multimodal Functions

The shifted & rotated multimodal functions f_{13} to f_{17} are particularly difficult to optimize because they comprise a huge number of local optima and their dimensions are non-separable. Table 6 shows that none of the four algorithms succeeds in finding the global optima of these functions. Nevertheless, TEA still obtains the best or the second best average results on four of the five functions. Meanwhile, as reported by Table 7, TEA is the only algorithm that finds acceptable solutions with over 80% successful rates on all the five functions. All the above results show that even in complex problems, TEA can maintain certain advantages in terms of solution accuracy, search speed, and reliability. These advantages can also be observed from the convergence curves depicted in Figure 5.

Table 6. Comparison of Solution Accuracy on Shifted & Rotated Functions[1]

Func	Alg	Best	Mean	Std.	Significance[2]
f_{13}	EPLS	**20.04**	**20.25**	**0.09444**	−1
	CLPSO	20.85	20.95	0.04492	+1
	JADE	20.77	20.93	0.04796	+1
	TEA	20.14	20.29	0.06480	-
f_{14}	EPLS	51.74	111.65	26.80	+1
	CLPSO	73.78	122.02	18.53	+1
	JADE	**22.52**	**33.58**	**4.483**	−1
	TEA	26.86	59.76	13.70	-
f_{15}	EPLS	22.51	29.85	3.689	+1
	CLPSO	22.30	26.12	1.705	+1
	JADE	26.85	32.91	3.622	+1
	TEA	**10.29**	**15.86**	**2.062**	-
f_{16}	EPLS	1.437	3.708	1.674	−1
	CLPSO	**0.5345**	**1.327**	**0.413**	−1
	JADE	1.312	1.572	0.124	−1
	TEA	2.309	5.587	1.967	-
f_{17}	EPLS	12.45	13.31	0.3259	+1
	CLPSO	12.03	12.88	0.1998	+1
	JADE	12.98	13.26	0.1385	+1
	TEA	**11.64**	**12.54**	**0.3404**	-

[1] The results marked in **boldface** are the best among the four algorithms.
[2] '+1', '−1', and '0' in the significance column indicate that compared to the result of TEA, the annotated result is significantly worse, significantly better, or insignificantly different, respectively.

Table 7. Comparison of Search Speed & Convergence Reliability[1]

Func	Average FEs Number				Successful Rate (%)			
	EPLS	CLPSO	JADE	TEA	EPLS	CLPSO	JADE	TEA
f_{13}	**14651**	95355	183811	48703	**100**	86	82	**100**
f_{14}	**25355**	242946	134641	32973	32	13	**100**	**100**
f_{15}	-	-	-	**27708**	0	0	0	**100**
f_{16}	**13781**	86337	34305	95808	98	**100**	**100**	98
f_{17}	**23489**	168756	203415	49684	16	76	8	86

[1] The results marked in **boldface** are the best among the four algorithms.

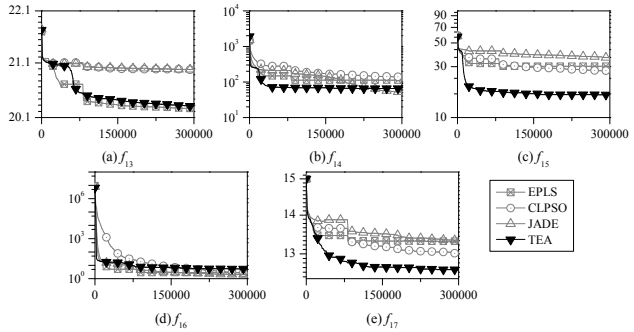

Figure 5. Evolution curves of EPLS, CLPSO, JADE, and TEA on shifted & rotated multimodal functions f12 to f17.

4. CONCLUSION

A new EC mechanism called TEA has been developed from the evolution mechanism of tribal ecosystems. An essential feature that differentiates TEA from other EC algorithms is that it uses the concept of tribes to extend the basic evolution unit from discrete solution points to solution sets in different subareas of the search space. Specifically, a tribe in TEA describes the solution set in a particular subarea with the locally best-so-far solution analogous to the tribal chief and the radius of the subarea analogous to the tribal diversity. The search experience in the subarea is also recorded as the self-advancing history of the tribe. Simulating tribal ecosystems, four evolution operators are designed for evolving tribes from roughly estimated promising subareas to the precise location of the global optimum.

The promising results of TEA on a diverse set of benchmark functions reveal the significance of our research in twofold. First, in terms of algorithmic design, we show that extending the basic evolution unit from solution points to solution sets in different subareas in the search space is beneficial for improving the performance of EC algorithms. This is because the extension allows more useful information to be involved in the evolution and thus facilitates the use of such information for helping the search procedure. Encouraged by the promising performance of TEA, it is worthwhile to investigate how the basic evolution unit can be further extended for incorporating more useful information efficiently. Second, in terms of performance, TEA provides an effective way for finding high-quality solutions at fast speed. Experimental results show that TEA has a general advantage against a number of representative EC algorithms, especially in terms of the search speed. It will be interesting to study the performance of TEA in practical problems.

5. ACKNOWLEDGEMENTS

This work was supported in part by the National High-Technology Research and Development Program (863 Program) of China No.2013AA01A212, in part by the NSFC for Distinguished Young Scholars 61125205, in part by the NSFC No. 61332002, No. 61300044 and No. 61170220.

6. REFERENCES

[1] J. Zhang, H. S. –H. Chung, and W. –L. Lo, "Clustering-based adaptive crossover and mutation probabilities for genetic algorithms," *IEEE Trans. Evol. Comput*, vol. 11, no. 3, pp. 326 – 335, June 2007.

[2] Z. –H. Zhan, J. Zhang, Y. Li, and H. S. –H. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst. Man. Cyber. – B*, vol. 39, no. 6, pp. 1362 – 1381, Dec. 2009.

[3] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945 – 959, Oct. 2009.

[4] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evolutionary Computation*, vol.10, no.3, pp.281-295, Jun. 2006.

[5] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 55 – 66, Feb. 2011.

[6] G. Guo and S. Yu, "Evolutionary Parallel Local Search for Function Optimization," *IEEE Trans. Syst. Man. Cyber. – B*, vol. 33, no. 6, pp. 864 – 876, Dec. 2003.

[7] X. Li, "Niching without niching parameters: particle swarm optimization using a ring topology," *IEEE Trans. Evolutionary Computation*, vol. 14, no. 1, pp. 150-169, Feb. 2010.

[8] X. Chen, Y. –W. Ong, M. –H. Lim, and K. C. Tan, "A multi-facet survey on memetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 15, no. 5, pp. 591 – 607, Oct. 2011.

[9] Y.-S. Ong, M.-H. Lim, N. Zhu, and K.-W. Wong, "Classification of adaptive memetic algorithms: a comparative study," *IEEE Trans. Syst. Man. Cyber. – B*, vol. 36, no. 1, pp. 141 – 152, Feb. 2006.

[10] T. Park and K. R. Ryu, "A dual-population genetic algorithm for diversity control," *IEEE Trans. Evol. Comput.*, vol. 14, no. 6, pp. 865 – 884, Dec. 2010.

[11] H. Kwasnicka and M. Przewozniczek, "Multi-population pattern searching algorithm: A new evolutionary method based on the idea of messy genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 15, no. 5, pp. 715 – 732, Oct. 2011.

[12] M. H. Fried, *The Notion of Tribe*, California: Cummings Publishing Company, 1975.

[13] fA. Hedayat, N. Sloane, and J. Stufken, *Orthogonal Arrays, Theory and Applications*. New York: Springer-Verlag, 1999.

[14] B.S. Everitt, S. Landau,and M. Leese, *Cluster Analysis* (4th Edition), Oxford University Press, 2001.

[15] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evolutionary Computation*, vol.3, no.2, pp.82-102, Jul. 1999.

[16] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwarki, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," *Technical Report, Nanyang Technological University, Singapore, and KanGAL Report Number 2005005*, May 2005.