Polygene-based Evolution: A Novel Framework for Evolutionary Algorithms

Shuaiqiang Wang[1,2], Byron J. Gao[2], Shuangling Wang[3], Guibao Cao[3], Yilong Yin[3*]
[1]Shandong University of Finance and Economics, Jinan, 250014 China
[2]Texas State University-San Marcos, San Marcos, TX 78666, USA
[3]Shandong University, Jinan, 250101 China
swang@sdufe.edu.cn, bgao@txstate.edu,
{slwang, gbcao}@mail.sdu.edu.cn, ylyin@sdu.edu.cn

ABSTRACT

In this paper, we introduce polygene-based evolution, a novel framework for evolutionary algorithms (EAs) that features distinctive operations in the evolution process. In traditional EAs, the primitive evolution unit is *gene*, where genes are independent components during evolution. In polygene-based evolutionary algorithms (PGEAs), the evolution unit is *polygene*, i.e., a set of co-regulated genes. Discovering and maintaining quality polygenes can play an effective role in evolving quality individuals. Polygenes generalize genes, and PGEAs generalize EAs. Implementing the PGEA framework involves three phases: polygene discovery, polygene planting, and polygene-compatible evolution. Extensive experiments on function optimization benchmarks in comparison with the conventional and state-of-the-art EAs demonstrate the potential of the approach in accuracy and efficiency improvement.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database Applications–*Data mining*

General Terms: Algorithms, Experimentation, Performance.

Keywords: Polygene, Evolutionary algorithms, Optimization, Data mining

1. INTRODUCTION

Evolutionary algorithms (EAs) [4] was derived from Darwinian evolutionary principles and widely applied in computationally difficult optimization and classification problems [2, 6]. In conventional EAs, the primitive evolution unit is gene, where genes are independent components during evolution.

In this paper, we introduce a novel polygene-based evolution framework. In polygene-based evolutionary algorithms (PGEAs), the primitive evolution unit is polygene. We consider a group of genes co-regulated and forming a polygene if they frequently occur in a population. As polygenes generalize genes, PGEA algorithms generalize EAs.

In biology, a *trait*, such as height, eye color or body mass, is a distinct variant of a phenotypic character of an organism. A *polygene* is a group of nonallelic genes that together influence a phenotypic trait [5]. Over evolutionary time, organisms evolve adaptive traits that enable them to survive [3].

In EAs, we do not have explicit, observable traits. However, inspired by the biological phenomenon, we speculate that there can be polygenes, each being a group of co-regulated genes, that have direct influence to the goodness of individuals. Thus, in the PGEA framework, we discover and maintain those quality polygenes throughout the evolution process in order to evolve quality individuals.

Using polygene instead of gene as the primitive evolution unit can significantly reduce the search space while retaining quality individuals. For optimization problems, a larger search space contains more feasible solutions and may lead to better results if the search is exhaustive. However, search heuristics such as EAs only search a very small part of the search space. PGEAs allow search to be done within a significantly reduced space that retains quality individuals, having increased probability to find good solutions.

In principle, PGEAs are highly promising as a new optimization tool. Implementation can also play a critical role in materializing the potential of the framework. In this study, we use a three-phase approach: (I) polygene discovery, (II) polygene planting, and (III) polygene-compatible evolution. For Phase I, we adopt an associative classification-based approach to discover quality polygenes. For Phase II, we perform probabilistic planting to maintain the diversity of individuals. For Phase III, we incorporate polygene-compatible crossover and mutation in producing next generation of individuals.

2. BACKGROUND

Association rules. The association rule mining problem [1] was first introduced in the context of mining transaction databases.

Let A be a set including all items. A set $X = \{i_1, i_2, \ldots, i_k\} \subseteq A$ is called an *itemset*, or k-itemset because X contains k items. A *transaction* over A is a tuple $T = (tid, I)$ where tid is the transaction identifier and I is an itemset. T is said to support X if $X \subseteq I$. A transaction database D over A is a set of transactions over A.

The *support* of an itemset X in D is the probability that X occurs in D, which is estimated by the proportion of the transactions in D containing X, formally, $\mathsf{sup}(X) = \frac{|\{T|T=(tid,I)\in D \wedge X \subseteq I\}|}{|D|}$.

An *association rule* is an implication $X \Rightarrow Y$, where $X, Y \subseteq A$ and $X \cap Y = \varnothing$. The support of $X \Rightarrow Y$ is calculated as the probability that a transaction in D contains both X and Y, formally, $\mathsf{sup}(X \Rightarrow Y) = \mathsf{sup}(X \cap Y)$. The confidence of $X \Rightarrow Y$ is calculated as the conditional probability that a transaction having

*Contact author.

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

X also contains Y, formally, $\mathsf{conf}(X \Rightarrow Y) = Prob(Y|X) = \frac{\mathsf{sup}(X \cap Y)}{\mathsf{sup}(X)}$.

Classification based on associations (CBA). CBA is the first algorithm for associative classification with *ruleitems*. A ruleitem is a frequent and accurate association rule $X \Rightarrow y$, where X is an itemset, and $y \in Y$ is a class label. Let $Y = \{y_1, y_2, \ldots, y_n\}$ be the set of the class labels. According to the definitions of the association rules, the support of the ruleitem $X \Rightarrow y_1$ is calculated as $\mathsf{sup}(X \Rightarrow y_1) = \mathsf{sup}(X \cap \{y = y_1\})$, and the confidence is calculated as $\mathsf{conf}(X \Rightarrow y_1) = \frac{\mathsf{sup}(X \cap \{y=y_1\})}{\mathsf{sup}(X)}$. Particularly, k-ruleitem denote a ruleitem $X \Rightarrow y$ where X has k items. An ruleitem is *frequent* if its support is no less than a given minimum frequency threshold θ. An ruleitem is *accurate* if its confidence is no less than a given minimum accuracy threshold ϑ.

3. THE PGEA FRAMEWORK

3.1 Definitions

DEFINITION 1. *Gene. In an individual, a gene is a tuple (i, v) where i is the position of the gene in the individual, and $v \in \{0, 1\}$ is the value.*

An individual is a set of such tuples. Since the position information is redundant, for simplicity, an individual can also be represented as a sequence $\langle v_1 v_2 \ldots v_m \rangle$. For example, $\langle 0001101010 \rangle$ represents the same gene as $\{(1, 0), (2, 0), \ldots, (9, 1), (10, 0)\}$. In the literature, sometimes "chromosome" is used for "gene".

In the PGEA framework, we consider a group of genes co-regulated and forming a polygene if they frequently occur in a population.

DEFINITION 2. *Polygene. In a population $\mathcal{P}$, a set of genes $\alpha = \{(i_1, v_1), (i_2, v_2), \ldots, (i_l, v_l)\}$ form a polygene if α occurs $\geq \theta$ times in $\mathcal{P}$.*

For example, in individual $\langle 0\mathbf{0}01\mathbf{1}01010 \rangle$, suppose $\alpha = \{(2, 0), (5, 1)\}$ (the bold font genes) satisfies the frequency threshold, then α forms a polygene. A polygene with length $l = 1$ is a *singleton polygene*. Polygenes can overlap, i.e., a same gene may occur in two or more different polygenes.

We are particularly interested in the quality polygenes. A basic observation is that good individuals often contain many quality polygenes, medium individuals usually contain some of them, but bad individuals generally contain few of them. Thus, quality polygenes, referring to *elite polygenes* (*elites* for short), can be used to distinguish good individuals from bad ones.

DEFINITION 3. *Elite Polygene. Let $\mathcal{P}$ be the population set containing all possible individuals. Each individual in $\mathcal{P}$ can be labeled as* `high`*,* `medium` *and* `low` *respectively based on their fitness scores. This way, $\mathcal{P}$ can be divided into 3 subsets: high-quality $\mathcal{P}_h$, medium-quality $\mathcal{P}_m$, and low-quality $\mathcal{P}_l$. A polygene α is an elite polygene if the ruleitem $\alpha \Rightarrow$* `high` *is a class association rule for $\mathcal{P}_h \cup \mathcal{P}_l$, i.e., $\mathsf{sup}(\alpha \Rightarrow \texttt{high}) > \vartheta_s \wedge \mathsf{conf}(\alpha \Rightarrow \texttt{high}) > \vartheta_c$, where $0 < \vartheta_s, \vartheta_c < 1$.*

Unfortunately, cannot traverse all possible individuals in $\mathcal{P}$ to get $\mathcal{P}_h$ and $\mathcal{P}_l$ exactly, but can only use the current population to estimate them approximately. Due to the inaccuracy of the training data, the quality of the generated elites with associative classification cannot be guaranteed. Thus, a further verification is necessary for elite generation.

3.2 Overview

Our PGEA framework uses polygenes as the primitive evolution unit. Its evolution process is similar to that of EAs. After initialization, it iterates from generation to generation.

There are three phases within each generation: (I) polygene discovery, (II) polygene planting, and (III) polygene-compatible evolution. Algorithm 1 summarizes them in pseudocode. Line 1 performs initialization, i.e., randomly generating an initial population of individuals, evaluating their fitness scores, and selecting the best individual from them based on their fitness scores. Lines 2–6 show the iterative evolution process of PGEA from generation to generation. In particular, line 3 is for Phase I, polygene discovery; line 4 is for Phase II, polygene planting; line 5 is for Phase III, polygene-compatible evolution.

Algorithm 1: PGEA Framework

Input : fitness function f, termination criteria tc, planting probability ε
Output: best individual b

```
1 P_0 ←Initialize (); // initial population
2 repeat
3 |  E ←Mine (P_{g−1}); // polygene discovery
4 |  Plant (P_{g−1}, E, ε); // polygene planting
5 |  P_g ← Evolve (P_{g−1}); // polygene-compatible evol.
6 until tc are met
```

3.3 Phase I: Polygene Discovery

In Phase I, we mine polygenes using frequent pattern mining techniques and select elite ones with a CBA-based approach.

Database transformation. Let $\mathcal{P}$ be the current population containing N individuals. Let $\mathcal{P}_h$ be the best $\xi \times N$ individuals in $\mathcal{P}$ with highest fitness scores, where each is labeled as `high`, and $0 < \xi < 1$ is called *high-quality proportion*. Let $\mathcal{P}_l$ be the worst $\eta \times N$ individuals with lowest fitness scores in $\mathcal{P}$, where each is labeled as `low`, and $0 < \eta < 1$ is called *low-quality proportion*. Other individuals are labeled as `medium`.

For database transformation, each gene (i, v) in p is transformed into an item. Let A be the itemset containing all potential items. Since for conventional EA, there are m positions in each individual, and for each gene position there are 2 potential values 0 and 1, there are totally $2m$ items in A.

Each individual $p = \langle v_1 v_2 \ldots v_m \rangle$ is transformed into a transaction $X_p = (tid_p, I_p)$, where $I_p = \{(1, v_1), \ldots, (m, v_m)\}$. This way, the population $\mathcal{P}$ are transformed into a transaction database D, where each transaction is also labeled as `high`, `medium`, or `low`.

Polygene mining. According to Definition 2, polygenes P can be mined from the transaction database D with pattern mining. Since efficiency has remained as a major challenge for pattern mining, to improve efficiency, we effectively reduce the number of mined polygenes without loss of quality by adding constraints on polygenes. (1) Only the polygenes where genes represent the same variable are considered because these genes are directly related to each other in forming a value of a variable. (2) The longest length of the frequent patterns (polygenes) is controlled by a predetermined parameter δ in order to reveal the main interdependency between genes but ignore the trivial cases.

Candidate elite selection. We consider quality polygene frequently occurring in high-quality individuals but seldom in low-quality ones, and candidate elite polygenes can be selected with associative clas-

sification. In our framework, a CBA-based approach is adopted to generate candidate elites based on the mined polygenes P.

Let α be a polygene mined from the transaction database D. A ruleitem $\alpha \Rightarrow \texttt{high}$ is generated as a candidate elite polygene. Let D_{hl} be the transaction set containing all transactions with labels high and low in D. According to the definition 3, α is an elite if $\mathsf{sup}(\alpha \Rightarrow \texttt{high}) > \vartheta_s \wedge \mathsf{conf}(\alpha \Rightarrow \texttt{high}) > \vartheta_c$ based on D_{hl}, where $0 < \vartheta_s, \vartheta_c < 1$.

Elite verification. As mentioned in Section 3.1, a further verification is necessary for elite generation because of the inaccuracy of the training data. In our framework, a testing-based validation is adopted.

Let α be a candidate elite polygene. Let N be the number of individuals in the population. Let κ be the *sample proportion*, i.e., $\kappa \times N$ individuals are sampled for verification. Let $S = \{p_{s_1}, p_{s_2}, \ldots p_{s_{\kappa \times N}}\}$ be the set of sampled individuals, where each individual p_{s_i} is associated with a fitness score $f(p_{s_i})$. After planting α into S (see Section 3.4), S is transformed into $S^\alpha = \{p^\alpha_{s_1}, p^\alpha_{s_2}, \ldots p^\alpha_{s_{\kappa \times N}}\}$. Polygene α is verified as an elite if better individuals can be generated after planting, formally, $\Delta f(S, \alpha) = \sum_{i=1}^{\kappa \times N} f(p^\alpha_{s_i}) - \sum_{i=1}^{\kappa \times N} f(p_{s_i}) < 0$.

3.4 Phase II: Polygene Planting

In Phase II, we plant the discovered quality polygenes to the current population to "magnify" their presence for their survival from evolution operations. For diversity, we adopt a probabilistic planting approach, where each polygene is selected for planting with a probability ε, called *planting probability*.

Let $\alpha = \{(i_1, v_1), (i_2, v_2), \ldots, (i_l, v_l)\}$ be the selected polygene. Let p be an individual. The planting process of α in p is to convert the gene values of p at position $i_1, i_2, \ldots, i_l$ into $v_1, v_2, \ldots, v_l$ respectively if a random generated number is less than ε.

EXAMPLE 1. *Let individual $p = \langle 00000101 \rangle$ be represented with 8 binary genes. Let $\alpha = \{(6, 0), (8, 0)\}$ be an elite polygene for planting. After planting, the individual becomes $p' = \langle 00000\mathbf{0}\mathbf{0}0 \rangle$.*

3.5 Phase III: Polygene-Compatible Evolution

For Phase III, polygene-compatible crossover and mutation are incorporated, where polygene is the unit of operation.

For polygene-compatible single-point crossover, the most important issue is to guarantee that polygenes in parent individuals can be inherited as a whole by offsprings without being broken.

DEFINITION 4. *Polygene-Compatible Crossover. A crossover for a pair of individuals p_1 and p_2 is polygene-compatible if their polygenes can be inherited as a whole in their offspring individuals p'_1 and p'_2.*

EXAMPLE 2. *Let p_1 and p_2 be 2 individuals and they contain polygenes $\alpha = \{(2, 0), (5, 1)\}$ and $\beta = \{(3, 1), (5, 0), (6, 0)\}$. The crossover point can be selected at position 7 (genes in italic font). As a result, offsprings p'_1 (inheriting α) and p'_2 (inheriting β) can be generated by crossover.*

$$p_1 = \langle \overbrace{00110}\ \overbrace{\mathit{1}010} \rangle \quad \Longrightarrow \quad p'_1 = \langle \overbrace{00110}\ \overbrace{\mathit{0}100} \rangle$$
$$p_2 = \langle \overbrace{01100}\ \overbrace{\mathit{0}100} \rangle \qquad\qquad p'_2 = \langle \overbrace{01100}\ \overbrace{\mathit{1}010} \rangle$$

Mutation of a polygene can be defined in various ways, where 1 or more genes in a polygene can be mutated. In our study, all genes in the polygene are mutated simultaneously during a polygene-compatible mutation operation.

Table 1: Properties of The Benchmarks

Property	F_1	F_2	F_3	F_4	F_5	F_6
Multi-model	✗	✗	✓	✓	✓	✓
Shifted	✓	✓	✓	✓	✓	✓
Non-separable	✗	✓	✓	✓	✓	✓
Multi-minimum	✗	✓	✓	✓	✓	✓
$\mathbf{x}^*$	$\mathbf{o}$	$\mathbf{o}$	$\mathbf{o}+1$	$\mathbf{o}$	$\mathbf{o}$	$\mathbf{o}$
$F(\mathbf{x}^*)$	f_b	f_b	f_b	f_b	f_b	f_b

DEFINITION 5. *Polygene-Compatible Mutation. In a polygene-compatible mutation, each gene (i_j, v_j) in polygene $\alpha = \{(i_1, v_1), (i_2, v_2), \ldots, (i_l, v_l)\}$ is mutated to $(i_j, \overline{v_j})$.*

3.6 Discussion

Although the PGEAs proposed in this study are based on the binary representation, it is possible to implement PGEAs with non-binary representations such as real number and tree. For the real number representation, instead of considering (i, v) as an item, a mapping function $f : \mathcal{R} \rightarrow L$ can be used to map the infinite gene value set $\mathcal{R}$ into a finite discrete label set L, e.g., $L = \{\texttt{low}, \texttt{medium}, \texttt{high}\}$. Then, we consider $(i, f(v))$ as an item. For the tree representation, we can use frequent tree/graph pattern mining for the discovery of polygenes.

4. EXPERIMENTS

Comparison partner. We chose conventional EA as our main comparison partner. Our implementation of PGEAs was based on conventional EAs. A direct comparison of the two will provide valuable and irreplaceable insights.

In addition, we also used two state-of-the-art EAs including a diploid EA [6] and a classic EDA, named population-based incremental learning (PBIL) [2].

Benchmark functions. We performed experiments on functional optimization problems to validate the PGEA framework. The benchmark functions are listed as follows [7].

$F_1(\mathbf{x}) = \sum_{i=1}^{n} z_i^2 + f_b$, where $\mathbf{z} = \mathbf{x} - \mathbf{o}$.
$F_2(\mathbf{x}) = \max_{1 \le i \le n}\{|z_i|\} + f_b$, where $\mathbf{z} = \mathbf{x} - \mathbf{o}$.
$F_3(\mathbf{x}) = \sum_{i=1}^{n-1} \left(100(z_{i+1} - z_i^2)^2 + (1 - z_i)^2\right) + f_b$, where $\mathbf{z} = \mathbf{x} - \mathbf{o} + 1$.
$F_4(\mathbf{x}) = \sum_{i=1}^{n} \left(z_i^2 - 10\cos(2\pi z_i + 10)\right) + f_b$, where $\mathbf{z} = \mathbf{x} - \mathbf{o}$.
$F_5(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{n} z_i^2 - \prod_{i=1}^{n} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f_b$, where $\mathbf{z} = \mathbf{x} - \mathbf{o}$.
$F_6(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} z_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi z_i)\right) + 20 + e + f_b$, where $\mathbf{z} = \mathbf{x} - \mathbf{o}$.

Since most benchmark functions were proposed for EAs with real number representations, without loss of generality, the following restrictions were applied in our experiments: (1) According to traditional experimental settings for EAs with binary representation, each variable x_i was represented by 10 binary genes, i.e., the range of x_i was restricted to $(-5.12, 5.12)$. (2) In the benchmark functions, $\mathbf{o}$ and f_b were predetermined as $[1.28, \ldots, 1.28]$ and -2.56 respectively.

Table 1 shows the properties of these benchmark functions, including whether they are multi-model, shifted or non-separable functions, whether they have multiple local minimums, the global optimum point $\mathbf{x}^*$, and the global optimization value $F(\mathbf{x}^*)$.

Parameter settings. For each function, 30 variables were used. In each generation 100 individuals were maintained. PGEA ter-

Table 2: Error Values $E_i(x) = F_i(\mathbf{x})$ - $F_i(\mathbf{x}^*)$

Items	Algs	F_1	F_2	F_3	F_4	F_5	F_6
$\overline{E(\mathbf{x})}$	PGEA	0.00	0.00	245	27.4	0.00	0.03
	EA	2.29	1.82	308	25.9	1.04	2.51
	Diploid	0.00	0.00	255	40.9	0.01	0.03
	PBIL	0.02	0.00	156	22.8	0.00	0.09
$\delta_{E(\mathbf{x})}$	PGEA	0.00	0.00	90.8	7.78	0.03	0.00
	EA	0.26	0.47	110	6.85	0.01	0.04
	Diploid	0.00	0.00	148	10.5	0.02	0.00
	PBIL	0.03	0.00	120	5.16	0.01	0.08

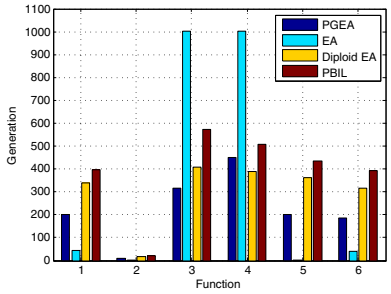


Figure 1: Number of generations before convergence.

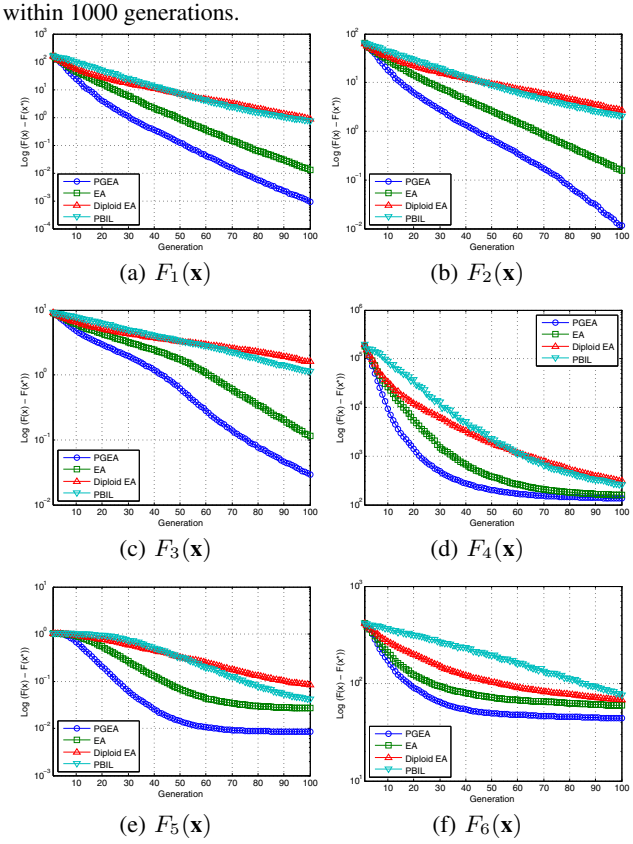
minated if the best solution cannot be improved for more than 50 generations or the maximum number of generations of 1000 was reached. These parameters were used for all algorithms in comparison.

The following parameter settings were used only for PGEA. In the polygene discovery phase, for each generation the best $\xi = 5\%$ and worst $\eta = 5\%$ individuals were considered as high-quality and low-quality individuals respectively. An itemset with maximum length $\delta = 2$ was frequent if its support was no less than $\theta = 50\%$. A ruleitem was a class association rule if its support and confidence were no less than $\vartheta_s = 30\%$ and $\vartheta_c = 70\%$ respectively. $\kappa = 5\%$ of individuals were sampled for elite verification. In polygene planting phase, each elite was selected for planting with a probability $\varepsilon = 10\%$.

Performance. We ran each algorithm 25 times. The results are summarized in Table 2 and Fig. 4. Table 2 presents the error values $E_i(\mathbf{x}) = F_i(\mathbf{x}) - F_i(\mathbf{x}^*)$ and their standard deviations for the benchmarks. From the table we can see that PGEA significantly gains in accuracy.

PGEA returned the most accurate optimization results for almost all benchmarks. For many functions, PGEA significantly outperformed the conventional EA. For example, for F_2, F_5 and F_6, the mean error values by the conventional EA are 2.2909, 1.0385 and 2.5053 while those by PGEA are 0.0016, 0.0097 and 0.0307 respectively. Besides, although PGEA ignored the polygenes crossing different variables (see Section 3.3), PGEA still significantly outperformed the conventional EA for 4 (F_2, F_3, F_5, F_6) out of 5 non-separable benchmark functions.

Fig. 2 shows the convergence rates within the first 100 generations. Fig. 4 shows the number of generations before convergence. From the figures we can see that PGEA significantly gains in efficiency. PGEA converged with the least number of generations for 5 out of 6 functions. Note that for F_1, F_2, F_5 and F_6, although the conventional EA converged the fastest, it failed to return reasonable results (see Table 2) and thus not considered. In general, PGEA converges significantly faster than the conventional EA. For example, for F_3 and F_4, PGEA took on average 316.56 and 450.6 generations to converge while conventional EA failed to converge within 1000 generations.



(a) $F_1(\mathbf{x})$ (b) $F_2(\mathbf{x})$

(c) $F_3(\mathbf{x})$ (d) $F_4(\mathbf{x})$

(e) $F_5(\mathbf{x})$ (f) $F_6(\mathbf{x})$

Figure 2: PGEA vs. comparison partners.

5. ACKNOWLEDGEMENT

This research was supported in part by the Natural Science Foundation of China (60970047, 61070097, 71171122), the Humanity and Social Science Foundation of Ministry of Education of China (12YJC630211), the Natural Science Foundation of Shandong Province of China (BS2012DX012), and the National Science Foundation (OCI-1062439, CNS-1058724).

6. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of SIGMOD'93*, 1993.

[2] S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical report, Carnegie Mellon University, 1994.

[3] P. J. Beurton, R. Falk, and H.-J. Rheinberger. *The Concept of the Gene in Development and Evolution*. Cambridge University Press, UK, 2000.

[4] J. Holland. *Adaptation in Natural and Artificial Systems*. The MIT Press, Cambridge, MA, 1975.

[5] E. Lawrence. *Henderson's Dictionary of Biology*. Pearson/Prentice Hal, New York, 2005.

[6] K. P. Ng and K. C. Wong. A new diploid scheme and dominance change mechanism for non-stationary function optimization. In *Proc. of GA'95*, 1995.

[7] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang. Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2007.