DOI: 10.1016/j.ins.2011.01.006

Polynomial modeling for time-varying systems based on a particle swarm optimization algorithm

[1*]**Kit Yan Chan,** [1]**Dillon S. Tharam and** [2]**Che Kit Kwong**

[1]Digital Ecosystem and Business Intelligence Institute, Curtin University of Technology, WA 6102, Australia

[2]Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hung Hom, Hong Kong

Abstract – In this paper, an effective particle swarm optimization (PSO) is proposed for polynomial models for time varying systems. The basic operations of the proposed PSO are similar to those of the classical PSO except that elements of particles represent arithmetic operations and variables of time-varying models. The performance of the proposed PSO is evaluated by polynomial modeling based on various sets of time-invariant and time-varying data. Results of polynomial modeling in time-varying systems show that the proposed PSO outperforms commonly used modeling methods which have been developed for solving dynamic optimization problems including genetic programming (GP) and dynamic GP. An analysis of the diversity of individuals of populations in the proposed PSO and GP reveals why the proposed PSO obtains better results than those obtained by GP.

Keywords Particle swarm optimization, time-varying systems, polynomial modeling, genetic programming

[*] Kit Yan Chan is the corresponding author. His email address is Kit.Chan@curtin.edu.au

1. Introduction

Genetic programming (GP) [25, 26] is a commonly used evolutionary computation method which is used to generate polynomial models for various systems such as chemical plants [38], time series systems [21], nonlinear dynamic systems [56], object classification systems [1, 65], machine learning systems [27], feature selection systems [43], object detection systems [37], speech recognition systems [11], control systems [5] and mechatronic systems [61]. The GP starts by creating a random initial population of individuals, each of which represents the structure of a polynomial model. Evolution of individuals takes place by mutation and crossover over generations, and individuals with high goodness-of-fit are selected as survivors in the next generation. The evolutionary process continues until the diversity of individuals of a population saturates to a low level or no progress can be found.

Observations reveal that polynomial models represented by individuals in the GP are distinct from each other in early generations. As the GP is progressing, polynomial models represented by individuals converge to a form, which achieves relatively higher goodness-of-fit in the population. Vaessens *et al.* [59] and Reeves [55] put this population-based optimization method into the context of local searches. Maintaining population diversity in GP is a key to preventing premature convergence and stagnation in local optima [17, 40]. Using GP, it is difficult to develop optimal polynomial models for time-varying systems whose structures or coefficients vary over time while the diversity of individuals in a population is low. Time-varying characteristics can commonly be found in many industrial systems [6, 22, 41, 44, 57, 66, 36, 34]. To develop models for time-varying environments, Wagner et al. [60] developed a GP approach in

which a varying window for capturing significant time series is proposed to generate time series models based on time series data. This approach cannot be applied for generating models for time-varying systems if the nature of the data is not all in time series formats. While mechanisms implemented on evolutionary algorithms have been well studied for solving various dynamic optimization problems [64], those implemented in GP have not been thoroughly studied for the development of polynomial models in time-varying environments. It is essential that an effective algorithm be developed for generating models that deal with time-varying characteristics, given their occurrence in many industrial systems.

Another more recent population based optimization method, particle swarm optimization (PSO) [15], inspires the movements of a population of individuals seeking optimal solutions. The movement of each individual is based on its best position recorded so far from previous generations and the position of the best individual among all the individuals [28, 29]. The diversity of the individuals can be maintained by selecting PSO parameters which provide a balance between global exploration, based on the position of the best individual in the swarm, and local exploration based on each individual's best previous position. Each individual can move gradually toward both its best position recorded to date and the position of the best individual in the population. Kennedy and Eberhart [29] demonstrated that PSO can solve many difficult optimization problems with satisfactory results. PSO outperforms evolutionary computation methods for solving various static optimization problems [13, 31, 53, 62], and various dynamic optimization problems [2, 9, 8, 52] in which the optima or landscapes of the problems vary over time. Although PSO can obtain satisfactory results when solving various dynamic optimization

problems, PSO has not currently been used on polynomial modelling for time-varying systems. The development of PSO for polynomial modelling for systems with time-varying characteristics is a new research area.

In this paper, a PSO is proposed for the development of polynomial models for time-varying systems in which the system coefficients vary over time. The basic operations of the proposed PSO are identical to those of the classical PSO [12] except that the elements of individuals are represented by arithmetic operations and system variables of polynomial models. The representation of elements takes the form of grammatical swarm [47, 48] or grammatical evolution [46]. The performance of the proposed PSO in the present paper is evaluated by developing models based on several sets of time-varying data which are generated based on time-varying functions with different time varying characteristics. In order to provide a comprehensive evaluation, a comparison is conducted of the results obtained by the proposed PSO with:

(a) classical GP [46] - in which the representation of individuals of population is identical to the one used in the proposed PSO;

(b) dynamic GP - which is integrated with a recent mechanism [63] for solving dynamic optimization problems;

(c) dynamic PSO - which is integrated with a recent mechanism [2] for dynamic optimization problems.

Even if additional computational effort is used in the dynamic PSO to maintain the diversity of individuals, no significant difference in diversity can be found between the proposed PSO and the dynamic PSO. Compared with the two GPs, the results indicate that the proposed PSO outperforms both classical GP and dynamic GP in developing

polynomial models for systems with both time-invariant data and time-varying data. The results can be explained by the diversity of individuals in the proposed PSO, which can be maintained in both early and later generations. The individuals of the proposed PSO continue to explore the solution spaces over the generations. In contrast the individuals of both the GP methods start to converge and get stuck on a solution after early generations.

This paper is organized as follows. Section 2 presents the operations of the proposed PSO. The experimental set-up for testing the proposed PSO, and the data sets used for evaluating the proposed PSO are presented in Sections 3.1 and 3.2 respectively. The experimental results and the analysis of the experimental results are presented in Section 3.3 and 3.4 respectively. Finally, conclusions and suggestions for further work are given in Section 4.

2. Particle swarm optimization

A time-varying system can be formulated as follows:

$$y = f^{t}(x_1, x_2, \ldots x_m) \tag{1}$$

where y is the output response, x_j, j=1,2,…m, is the j-th variable of the time-varying system, and f^{t} is the functional relationship of the time-varying system at time t. Based on a set of data which represents relations between the output response y and the variables, x_1, x_2, ..., x_m at time t, the time-varying system f^{t} in (1) can be generated in a polynomial form with constant coefficients at time t. The data set at time t is defined by $\mathbf{D}(t) = \left\{\left(\mathbf{x}^{D}(t,i), y^{D}(t,i)\right)\right\}_{i=1}^{N_D}$, where the corresponding values of the i-th data at time t is $\mathbf{x}^{D}(t,i) = \left(x_1{}^{D}(t,i), x_2{}^{D}(t,i), \ldots, x_m{}^{D}(t,i)\right) \in R^m$ and the corresponding value of the response output of the i-th data at time t is $y^{D}(t,i) \in R$. Where $\mathbf{D}(t)$ is available, f^{t} can be

generated as the high-order high-dimensional Kolmogorov-Gabor polynomial in expression (2):

$$y = a_0(t) + \sum_{i_1=1}^{m} a_{i_1}(t) x_{i_1} + \sum_{i_1=1}^{m}\sum_{i_2=1}^{m} a_{i_1 i_2}(t) x_{i_1} x_{i_2} + \sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\sum_{i_3=1}^{m} a_{i_1 i_2 i_3}(t) x_{i_1} x_{i_2} x_{i_3} + \ldots a_{123\ldots m}(t) \prod_{i_m=1}^{m} x_{i_m} \quad (2)$$

where $a_0(t)$, $a_1(t)$, $a_2(t)$, ..., $a_m(t)$, $a_{11}(t)$, $a_{12}(t)$, ..., $a_{mm}(t)$,... and $a_{mm\ldots m}(t)$ are the polynomial coefficients at time *t*. Equation (2) is a universal format of the polynomial model if the number of terms in equation (2) is large enough [18]. In this paper, a PSO is proposed in order to generate the time-varying model at time *t* based on equation (2), using an available set of data at time *t*. Based on [12], the proposed PSO uses a number of individuals, which constitute a swarm, and each individual represents a time-varying model. Each individual traverses the search space to trace the polynomial model of the time-varying system whose system coefficients vary over time.

In the PSO, each individual is represented by the system variables (x_1, x_2, …, and x_m) and the arithmetic operations ('+', '-' and '*') of the system model as defined in (2). *m* is the number of variables of the system model. A similar mechanism was first proposed by Kennedy and Eberhart [30] for representing discrete binary variables, and has been applied to the PSO for solving flowshop scheduling problems [31, 51, 58]. The *i*-th individual at generation *g* is defined as $P_i^g = \left(p_{i,1}^g, p_{i,2}^g, \ldots, p_{i,N_p}^g \right)$; where $N_p > m$; $i = 1,2,\ldots, N_{pop}$; N_{pop} is the number of individuals of the swarm; N_p is the number of elements of the individual; and N_{pop} is an odd number; $p_{i,k}^g$ is the *k*-th element of the *i*-th individual at the *g*-th generation, and $p_{i,k}^g$ is in the range between 0 to 1 i.e. $p_{i,k}^g \in \{0\ldots1\}$.

If the value of N_p is large, a larger number of terms can be generated in the model, and the model can better fit the data which is used for model development. However, a model may contain too many unnecessary and complex terms. A complex, over-parameterized model with a large number of parametric terms reduces the transparency and ease of interpretation of the model leading to overfitting problems. To prevent the PSO from generating models which are too complex, the value of N_p has to be selected carefully. The value of N_p can be determined based on the trial and error method, and the value of N_p cannot be set too high, otherwise redundant terms can be produced. If the number of variables of the system model is 4, N_p can be initially set as 10. If the modelling error obtained by the PSO is not satisfactory, the value of N_p can be increased until a satisfactory modelling error is achieved. If the modelling error obtained by the PSO is satisfactory, the value of N_p can be decreased until just before an unsatisfactory modelling error is achieved.

The elements in odd numbers (i.e. $p_{i,1}^g$, $p_{i,3}^g$, $p_{i,5}^g$,...) are used to represent the system variables, and the elements in even numbers (i.e. $p_{i,2}^g$, $p_{i,4}^g$, $p_{i,6}^g$,...) are used to represent the arithmetic operations. For odd k, if $0 < p_{i,k}^g \le \frac{1}{m+1}$, no system variable is represented by the element $p_{i,k}^g$. If $\frac{l}{m+1} < p_{i,k}^g \le \frac{l+1}{m+1}$ with l>0, $p_{i,k}^g$ represents the *l-th* system variable, x_l. System variables represented by the individual are summarized in Table 1.

Table 1: Representation of system variables

The *k*-th element	$0 < p_{i,k}^g \le \frac{1}{m+1}$	$\frac{1}{m+1} < p_{i,k}^g \le \frac{2}{m+1}$	$\frac{2}{m+1} < p_{i,k}^g \le \frac{3}{m+1}$	….	$\frac{m}{m+1} < p_{i,k}^g \le 1$
The system variable	No system variable	x_1	x_2	….	x_m

*k is an odd number

In the polynomial model, '+', '-' and '*' are the only three arithmetic operations considered. For even k, if $0 < p_{i,k}^g \le \frac{1}{3}$, $\frac{1}{3} < p_{i,k}^g \le \frac{2}{3}$ and $\frac{2}{3} < p_{i,k}^g \le 1$, the element $p_{i,k}^g$ represents the arithmetic operations '+', '-' and '*' respectively. Arithmetic operations represented by the individual are summarized in Table 2.

Table 2: Representation of arithmetic operations

The *k*-th element	$0 < p_{i,k}^g \le \frac{1}{3}$	$\frac{1}{3} < p_{i,k}^g \le \frac{2}{3}$	$\frac{2}{3} < p_{i,k}^g \le 1$
The arithmetic operations	+	-	*

*k is an even number

For example, the i-th individual at generation g with 11 elements is used to represent a polynomial model of the time-varying system at time t, which consists of 4 system variables (i.e. x_1, x_2, x_3 and x_4):

$p_{i,1}^g$	$p_{i,2}^g$	$p_{i,3}^g$	$p_{i,4}^g$	$p_{i,5}^g$	$p_{i,6}^g$	$p_{i,7}^g$	$p_{i,8}^g$	$p_{i,9}^g$	$p_{i,10}^g$	$p_{i,11}^g$
0.18	0.41	0.94	0.92	0.41	0.89	0.06	0.35	0.81	0.01	0.74

The elements in the individual are within the following ranges:

$p_{i,1}^g$	$p_{i,2}^g$	$p_{i,3}^g$	$p_{i,4}^g$	$p_{i,5}^g$	$p_{i,6}^g$	$p_{i,7}^g$	$p_{i,8}^g$	$p_{i,9}^g$	$p_{i,10}^g$	$p_{i,11}^g$
0< 0.18 ≤ 1/5	1/3< 0.41 ≤ 2/3	4/5< 0.94 ≤5/5	2/3< 0.92 ≤3/3	2/5< 0.41 ≤3/5	0< 0.08 ≤1/3	0< 0.06 ≤1/5	0< 0.35 ≤1/3	4/5< 0.81 ≤5/5	0< 0.01 ≤1/3	3/5< 0.74 ≤4/5

Therefore, the model is represented in the following form:

$p_{i,1}^g$	$p_{i,2}^g$	$p_{i,3}^g$	$p_{i,4}^g$	$p_{i,5}^g$	$p_{i,6}^g$	$p_{i,7}^g$	$p_{i,8}^g$	$p_{i,9}^g$	$p_{i,10}^g$	$p_{i,11}^g$
0	-	x_4	*	x_2	+	0	+	x_4	+	x_3

which is equivalent to:

$$\varphi_i^g(\mathbf{x}) = 0 - x_4 \cdot x_2 + 0 + x_4 + x_3$$

or

$$\varphi_i^g(\mathbf{x}) = -x_4 \cdot x_2 + x_4 + x_3 .$$

The PSO is used only to find the structure of the polynomial and not the coefficients. The system coefficients $a_0(t)$, $a_1(t)$, $a_2(t)$ and $a_3(t)$ are determined after the structure of the time-varying model at time t is established, where the number of coefficients is 4. The completed time-varying model at time t is represented as follows:

$$f_i^g(\mathbf{x}) = a_0(t) - a_1(t) \cdot x_4 \cdot x_2 + a_2(t) \cdot x_4 + a_3(t) \cdot x_3$$

In this research, the system coefficients $a_0(t)$, $a_1(t)$, $a_2(t)$ and $a_3(t)$ are determined by the orthogonal least squares algorithm (OLSA) [2, 5], which has been demonstrated to be effective in determining system coefficients in polynomial models [39]. Details of the orthogonal least squares algorithm can be found in [3, 7].

The polynomial model represented by each individual is evaluated based on the root mean absolute error (RMAE). This reflects the differences between the predictions by the model of the time-varying system at time t and the actual values of the data sets at time t. The RMAE of the i-th individual at the g-th generation $RMAE_i^g$ can be calculated based on (3).

$$RMAE_i^g = 100\% \times \sqrt{\frac{1}{N_D}\sum_{j=1}^{N_D}\left|\frac{y^D(t,j) - f_i^g\left(\mathbf{x}^D(t,j)\right)}{y^D(t,j)}\right|}, \qquad (3)$$

where f_i^g is the polynomial model represented by the i-th individual P_i^g at the g-th generation, $\left(\mathbf{x}^D(t,j), y^D(t,j)\right)$ is the j-th data set at time t, and N_D is the number of training data sets used for developing the polynomial model of the time-varying system.

The velocity $v_{i,k}^g$ (corresponding to the flight velocity in a search space) and the k-th element of the i-th individual at the g-th generation $p_{i,k}^g$ are calculated by expressions (4) and (5) of the PSO [10] respectively:

$$v_{i,k}^g = K\left(v_{i,k}^{g-1} + \varphi_1 \cdot rand(\,) \cdot \left(pbest_{i,k} - p_{i,k}^{g-1}\right) + \varphi_2 \cdot rand(\,) \cdot \left(gbest_k - p_{i,k}^{g-1}\right)\right) \qquad (4)$$

$$p_{i,k}^g = p_{i,k}^{g-1} + v_{i,k}^g \qquad (5)$$

where

$$pbest_i = \left[\, pbest_{i,1}, \quad pbest_{i,2}, \quad \ldots \quad pbest_{i,N_p} \,\right],$$

$$gbest = \left[\, gbest_1, \quad gbest_2, \quad \ldots \quad gbest_{N_p} \,\right],$$

$k = 1,2, \ldots, N_p,$

The best previous position so far of an individual is recorded from the previous generation and is represented as $pbest_i$; the position of the best individual among all the individuals is represented as *gbest*; *rand*() returns a uniform random number in the range of [0,1]; w is an inertia weight factor; φ_1 and φ_2 are acceleration constants [13]; K is a constriction factor derived from the stability analysis of equation (4) to ensure that the

system converges, but not prematurely. K is a function of φ_1 and φ_2 as reflected in the following equation:

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \tag{6}$$

where $\varphi = \varphi_1 + \varphi_2$ and $\varphi > 4$.

The proposed PSO utilizes $pbest_i$ and *gbest* to modify the current search point to prevent the individuals from moving in the same direction, but to converge gradually toward $pbest_i$ and *gbest*. g is the current generation number, G is the total number of generations [14].

In (4), the particle velocity is limited by a maximum value $v_{\max}$. The parameter $v_{\max}$ determines the resolution with which regions are to be searched between the present position and the target position. This enhances the local exploration of a search process. If $v_{\max}$ is too high, individuals might fly past good solutions. If $v_{\max}$ is too small, individuals may not explore sufficiently beyond local solutions. $v_{\max}$ was often set as 10%–20% of the dynamic range of the element on each dimension. The pseudo code of the proposed PSO is presented in Figure 1.

{

$g \rightarrow 0$ // g is the generation number

Initialize a set of individuals $P_1^g, P_2^g, ..., P_{N_{pop}}^g$ // $P_i^g = \left(p_{i,1}^g, p_{i,2}^g, ..., p_{i,N_p}^g \right)$

Evaluate each individual P_i^g based on (3)

while (g<G) **do** // G is the total number of generations

{

$g \rightarrow g$+1

Update the velocity $v_{i,k}^g$ based on (4)

if $v_{i,k}^g > v_{max}$

$v_{i,k}^g = v_{max}$

end

if $v_{i,k}^g < -v_{max}$

$v_{i,k}^g = -v_{max}$

end

Update each element of each particle $p_{i,k}^g$ based on (5)

Evaluate each particle P_i^g based on (3)

Update $pbest_i$ and $gbest$

}

}

Figure 1: Pseudo code of the PSO

3. Polynomial modelling

In this section, the effectiveness of the PSO in modeling time-invariant or time-varying systems is evaluated based on both the time-invariant data and time-varying data. The PSO and the other commonly used, but recently developed, algorithms are compared.

3.1 Time-invariant and time-varying data

For time-invariant data, five sets of data, namely *static data,* Π_{Sph}^{stat}, Π_{Ros}^{stat}, Π_{Ras}^{stat}, Π_{Gri}^{stat} and Π_{Ack}^{stat}, were generated based on each of the five benchmark functions, Sphere, Griewank, Rastrigin, Rosenbrock and Ackley (see Table 3) by randomly choosing 100 numbers X_i in each of the predefined intervals $\left[X_{\min}, X_{\max}\right]^n$. The 100 corresponding output responses Y_i

are computed by the benchmark function $Y_i = F(X_i)$ whose landscape and optimum are static with respect to time. The dimension of each benchmark function is n=4. The Sphere (F_{Sph}) and Rosenbrock (F_{Ros}) functions are unimodal (a single local and global optimum), and the Griewank (F_{Gri}), Rastrigin (F_{Ras}), and Ackley (F_{Ack}) functions are multimodal (several local optima).

Table 3: Benchmark functions and initialization areas

Benchmark functions	Initialization areas $[X_{\min}, X_{\max}]^n$
Sphere (*Sph*): $F_{Sph}(\mathbf{x}) = \sum_{i=1}^{n} (x_i)^2$	$[-50,50]^n$
Rosenbrock (*Ros*): $F_{Ros}(\mathbf{x}) = \sum_{i=1}^{n-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right)$	$[-30,30]^n$
Rastrigin (*Ras*): $F_{Ras}(\mathbf{x}) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	$[-5.12,5.12]^n$
Griewank (*Gri*): $F_{Gri}(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600,600]^n$
Ackley (*Ack*): $F_{Ack}(\mathbf{x}) = -20 \cdot \exp\left(-0.2\sqrt{1/n \cdot \sum_{i=1}^{n} x_i^2}\right) - \exp\left(1/n \cdot \sum_{i=1}^{n} \cos(2\pi \cdot x_i)\right) + 20 + e$	$[-32,32]^n$

The time-varying data used in this study was generated by a set of time-varying functions which were extended from the benchmark functions shown in Table 3. X_i was generated by randomly chosen numbers in the predefined interval of the benchmark function. Y_i was computed in each generation of the PSO run by the time-varying function $Y_i = F(X_i, t)$ whose landscape or optima varies over time t. The mechanisms for

the development of the time-varying functions were based on the dynamic properties of step changes of optima, changes of locations of optima, and changes of the landscapes of the benchmark functions [33, 42].

For those based on the mechanism of step changes of optima, the time-varying data was generated based on each of the five benchmark functions, F_{Sph}, F_{Ros}, F_{Ras}, F_{Gri} and F_{Ack}. The optimum position $\mathbf{x}^+$ of each benchmark function is moved by adding or subtracting random values in all dimensions by a severity parameter *s*, at every change of the environment [32]. The choice of whether to add or subtract the severity parameter *s* on the optimum $\mathbf{x}^+$ is done randomly with an equal probability. The severity parameter *s* is defined by:

$$s = \begin{cases} +\Delta d \cdot (X_{\max} - X_{\min}) \text{ if rand}(\ ) \geq 0 \\ -\Delta d \cdot (X_{\max} - X_{\min}) \text{ if rand}(\ ) < 0 \end{cases}, \tag{7}$$

where Δd determines the scale of the step change of optima.

For each test run, a different random seed was used. The severity was chosen relative to the extension (in one dimension) of the initialization area of each benchmark function. The optima of the benchmark functions were periodically changed in every 100 generations of the runs of the algorithms. For small step changes of optima, $\Delta d = 5\%$ is selected, and five sets of time-varying data, (namely *step move data with* $\Delta d = 5\%$, Π_{Sph}^{Step-5}, Π_{Ros}^{Step-5}, Π_{Ras}^{Step-5}, Π_{Gri}^{Step-5} and Π_{Ack}^{Step-5}) were generated based on the five benchmark functions, F_{Sph}, F_{Ros}, F_{Ras}, F_{Gri} and F_{Ack}, in Table 3 respectively. For large step changes of optima, $\Delta d = 10\%$ is selected, and five sets of time varying data, (namely *step move data with* $\Delta d = 5\%$, $\Pi_{Sph}^{Step-10}$, $\Pi_{Ros}^{Step-10}$, $\Pi_{Ras}^{Step-10}$, $\Pi_{Gri}^{Step-10}$ and $\Pi_{Ack}^{Step-10}$) were generated respectively.

For those based on the mechanism of changes of locations of optima [21], the time-varying data was generated based on the following time-varying function:

$$F(x) = w(t) \cdot F_i(x) - (1.0 - w(t)) \cdot F_i(x + s) \quad (8)$$

where $w(t) = 1 - \frac{100}{G} \text{floor}\left(\frac{t}{100}\right)$, $0 \le t \le G$; G is the pre-defined number of generations; $F_i(x)$ is any of the five benchmark functions, F_{Sph}, F_{Ros}, F_{Ras}, F_{Gri} and F_{Ack}, in Table 3, s is a randomly chosen constant which is 20% of the range of the benchmark function $F_i(x)$. The optimum in $F(x)$ shifts from the original optimum $\mathbf{x}^+$ of $F_i(x)$ to the new optimum $(\mathbf{x}^+ + s)$ in every 100 generations. Based on these time-varying functions, five sets of time varying data, namely *shift data*, Π_{Sph}^{shift}, Π_{Ros}^{shift}, Π_{Ras}^{shift}, Π_{Gri}^{shift} and Π_{Ack}^{shift}, were generated based on the five benchmark functions, F_{Sph}, F_{Ros}, F_{Ras}, F_{Gri} and F_{Ack} in Table 3 respectively.

For those based on the mechanism of changes of the landscapes of the benchmark functions [24], the time-varying data was generated based on the following time-varying function, which is similar to equation (8):

$$F(x) = w(t) \cdot F_i(x + s) - (1.0 - w(t)) \cdot F_j(x) \quad (9)$$

where $F_i(x)$ is any of the five benchmark functions in Table 3, and $F_j(x)$ is another benchmark function. The landscape of $F(x)$ changes gradually from the landscape of $F_j(x)$ to the landscape of $F_i(x)$ in every 100 generations. s is a randomly chosen constant which is 20% of the range of the benchmark function $F_i(x)$. Based on the time-varying function (10), four sets of time varying data, namely *match data*, $\Pi_{Sph-Ros}^{match}$, $\Pi_{Sph-Ras}^{match}$, $\Pi_{Sph-Gri}^{match}$ and $\Pi_{Sph-Ack}^{match}$, were generated in which F_{Sph} is used as $F_j(x)$, and F_{Ros}, F_{Ras}, F_{Gri} or F_{Ack}

functions is used as $F_i(x)$. Another three sets of *match data*, $\Pi^{match}_{Ros-Ras}$, $\Pi^{match}_{Ros-Gri}$ and $\Pi^{match}_{Ros-Ack}$, were generated in which $F_{Ros}(x)$ is used as $F_j(x)$, and F_{Ras}, F_{Gri} or F_{Ack} is used as $F_i(x)$.

A brief summary of all the 27 data sets is presented in Table 4, and the benchmark functions, which can be used to generate the data sets, can be downloaded from the following link (http://www.4shared.com/account/dir/G2J--2eV/sharing.html).

Table 4: Description of the data sets

Data sets		Descriptions
Static data	Π^{stat}_{Sph}, Π^{stat}_{Ros}, Π^{stat}_{Ras}, Π^{stat}_{Gri}, Π^{stat}_{Ack}	The *static data* was generated by the benchmark functions, F_{Sph}, F_{Ros}, F_{Ras}, F_{Gri} and F_{Ack}.
Step move data with $\Delta d = 5\%$	Π^{Step-5}_{Sph}, Π^{Step-5}_{Ros}, Π^{Step-5}_{Ras}, Π^{Step-5}_{Gri}, Π^{Step-5}_{Ack}	The *step move data with* $\Delta d = 5\%$ were generated based on the benchmark functions, F_{Sph}, F_{Ros}, F_{Ras}, F_{Gri} and F_{Ack}, in which the locations of the optima change from $\mathbf{x}^+$ to $\mathbf{x}^+ + s$ in every 100 generations. s is 5% of the ranges of the benchmark functions.
Step move data with $\Delta d = 10\%$	$\Pi^{Step-10}_{Sph}$, $\Pi^{Step-10}_{Ros}$, $\Pi^{Step-10}_{Ras}$, $\Pi^{Step-10}_{Gri}$, $\Pi^{Step-10}_{Ack}$	The mechanism is the same as that for the above data sets except that s is 10% of the ranges of the benchmark functions.
Shift data	Π^{shift}_{Sph}, Π^{shift}_{Ros}, Π^{shift}_{Ras}, Π^{shift}_{Gri}, Π^{shift}_{Ack}	The *shift data* was generated based on the benchmark functions, F_{Sph}, F_{Ros}, F_{Ras}, F_{Gri} and F_{Ack}, in which the locations of the optima move from $\mathbf{x}^+$ to $\mathbf{x}^+ + s$ gradually. s is 20% of the ranges of the benchmark functions.
Match data based on F_{Sph}	$\Pi^{match}_{Sph-Ros}$, $\Pi^{match}_{Sph-Ras}$, $\Pi^{match}_{Sph-Gri}$, $\Pi^{match}_{Sph-Ack}$	The *match data based on* F_{Sph} was generated in which the landscape changes from F_{Sph} to F_{Ros}, F_{Ras}, F_{Gri} or F_{Ack}.
Match data based on F_{Ros}	$\Pi^{match}_{Ros-Ras}$, $\Pi^{match}_{Ros-Gri}$, $\Pi^{match}_{Ros-Ack}$	The *match data based on* F_{Ros} was generated in which the landscape changes from F_{Ros} to F_{Ras}, F_{Gri} or F_{Ack}.

3.2 Experiment Set-up

In this paper, because the basic operation of the PSO discussed in Section 2 is similar to classical PSO, it is called classical PSO, C-PSO in this paper. The following parameters, which can be found in reference [48], were implemented in the C-PSO: the number of particles in the swarm was 100; the number of elements in the particle was 30; both the acceleration constants φ_1 and φ_2 were set at 2.05; the maximum velocity $v_{\max}$ was 0.2; the pre-defined number of generations was 1000. Based on the results in [48], these parameters can produce satisfactory results when solving both parameterized and combinatorial problems. Therefore, these parameters are used in this research. The C-PSO was compared against the following five approaches for generating models based on both the time-invariant and time-varying data sets, which have been discussed in Section 3.1.

1. Classical genetic programming (C-GP): A commonly used method for polynomial modeling, the classical genetic programming (C-GP) [25, 26] was employed. Here the representation of the individuals of the grammatical genetic programming [46] is identical to the one of the representations of the C-PSO. The basic operations of the C-GP are shown in Figure 2 in the Appendix. The C-GP first starts by creating a random initial population $\Omega(g)$ of individuals $[\theta_1(g), \theta_2(g),\ldots \theta_{POP}(g)]$, while g=0. The i-th individual $\theta_i(g)$ at the g-th generation represents the structure of the time-varying model (2). For example, the i-th individual at the g-th generation $\theta_i(g)$ represents the structure of the following time-varying model at time t:

$$\theta_i(g) = x_1^2 - x_2^2 + x_1 \cdot x_2 \cdot x_4 \tag{10}$$

After determining the structure of the time-varying model $\theta_i(g)$, the system coefficients are determined. The completed time-varying model $\theta_i(g)$' is represented by:

$$\theta_i(g)' = a_0(t) + a_1(t)\cdot x_1^2 - a_2(t)\cdot x_2^2 + a_3(t)\cdot x_1\cdot x_2\cdot x_4 \quad (11)$$

where $a_0(t)$, $a_1(t)$, $a_2(t)$ and $a_3(t)$ are the system coefficients at time t, and are calculated by OLSA. This is the same as the one used in the C-PSO for calculating system coefficients. The classical genetic operations, point mutation and one-point crossover, were used. Standard roulette wheel selection was used. The following GA parameters were implemented in the C-GP: The population size is 100. The type of replacement is elitist. Crossover rate and mutation rate were 0.9 and 0.01 respectively. The pre-defined number of generations was 1000. The dimension of the individuals was 30.

2. Dynamic particle swarm optimization (D-PSO): D-PSO is identical to the C-PSO except for integration of the recent mechanisms for maintaining diversities of the swarms [2] when solving the dynamic optimization problem. The mechanism splits the whole set of particles into a set of interacting swarms. These swarms interact locally through an exclusion parameter and globally through an anti-convergence operator. Each swarm maintains its diversity by using either charged or quantum particles. Results show that when this mechanism for maintaining diversity in the PSO is used, the PSO outperforms the other PSO or evolutionary algorithms, even where they are integrated with other diversity maintaining mechanisms, for solving dynamic optimization problems. The performance of D-PSO was optimized by tuning

it with different settings for the number particles in the sub-swarms. 5, 10 and 25 particles in the sub-swarms were used and the best performance among them was recorded. The detailed description of the mechanisms used to maintain diversity in the swarms can be found in [2].

3. Dynamic genetic programming (D-GP): D-GP is identical to the C-GP except for integration of the recent mechanism [63] used for evolutionary algorithms on solving dynamic optimization problems. The mechanism relocates the positions of the individuals based on the changes of the landscape of the dynamic optimization problem and the average sensitivities of their decision variables to the corresponding change in the landscape. While integrating the mechanism in the evolutionary algorithm, the evolutionary algorithm outperforms the other dynamic evolutionary approaches for solving dynamic optimization problems. The detailed description of the mechanisms for maintaining diversity can be found in [63].

4. Polynomial-genetic algorithm (P-GA): P-GA is a genetic algorithm proposed by Potgieter and Engelbrecht [53] which can evolve structurally polynomial expressions in order to accurately describe a given data set. In P-GA, each individual is used to represent the structure of the polynomial and this is evolved based on the designed crossover and mutation operations. The coefficients of the polynomial are determined by OLSA [3, 7]. The crossover rate and the mutation rate were set at 0.1 and 0.2 respectively, which are the same as those used in [54]. The population size was set at 100. The individual length was set at 22.

5. Polynomial neural network (PNN): PNN is developed based on a genetic algorithm which is proposed by Oh and Pedrycz [50]. Individuals in the genetic algorithm are used to represent the parameters of the PNN including the number of input variables, the order of the polynomial and input variables, which lead to a structurally and parametrically optimized network. The coefficients of the polynomial are determined by OLSA [3, 7]. The number of layers of the PNN was set at 3. The crossover rate and the mutation rate used in the genetic algorithm were set at 0.65 and 0.1 respectively, which are the same as those used in [50]. The population size was set at 100. The individual length was set at 36.

3.3 Experimental results

Thirty runs were performed on the C-PSO, D-PSO, D-GP, C-GP, P-GA and PNN in generating polynomial models based on each of the 27 data sets shown in Table 4. In each generation of the runs, the RMAE obtained by the individuals of the six algorithms was recorded.

Online performance of the algorithms is demonstrated by the convergence plots. Figures 3a, 3b, 3c, 3d and 3e show the convergence plots for the step move data with $\Delta d = 5\%$ (Π_{Sph}^{Step-5}, Π_{Ros}^{Step-5}, Π_{Ras}^{Step-5}, Π_{Gri}^{Step-5}, Π_{Ack}^{Step-5}) respectively. It can be observed from Figures 3a-3e that the evolutionary algorithms, C-GP, D-GP, P-GA and PNN, converged more quickly in the early generations than did those of the PSO algorithms, C-PSO and D-PSO. However, the PSO algorithms, C-PSO and D-PSO, kept progressing after the early generations. Finally, both the PSO algorithms, C-PSO and D-PSO, reached a

smaller RMAE than that reached by the evolutionary algorithms, C-GP, D-GP, P-GA and PNN, in the final stage of the search. D-PSO can reach the smallest RMAE compared with those obtained by the other algorithms. Therefore in general, the PSO algorithms outperform the evolutionary algorithms in generating the models for these static data sets in later generations. For the rest of the data (static data, step move data with $\Delta d = 10\%$, shift data, match data based on F_{Sph}, match data based on F_{Ros}), a similar finding can be observed in that the convergence speed of the evolutionary algorithms was faster than that of the particle swarm optimization algorithms in the early generations. In the late generations, the particle swarm optimization algorithms can reach a smaller RMAE than that reached by the evolutionary algorithms.

The smallest RMEA among all generations of each run of each algorithm was recorded, and was averaged. This measure is called offline performance. The commonly used method for testing the significance of the results, the Wilcoxon Rank Sum Test, was used to compare the results between the two algorithms [19]. The results of the 30 runs for two algorithms form two independent random samples X and Y. The distributions of X and Y, F_X and F_Y, are compared using the null-hypothesis H_0: $F_X=F_Y$ and the one-sided alternative H_1: $F_X<F_Y$. We performed the significance tests at a significance level α = 0.01. Only if the probability of the observed difference is less than α, is the null-hypothesis rejected and the alternative hypothesis accepted. The significance comparison among a set of the six algorithms, C-PSO, D-PSO, D-GP, C-GP, P-GA and PNN, is displayed using a 6×6 matrix, where 'X' denotes that the result obtained by algorithm i and that obtained by algorithm j is statistically significant different. i and j are the position in the corresponding result table. An entry of '_' indicates that the result

obtained by algorithm *i* and algorithm *j* is not a statistically significant difference. We name such a matrix a *significance matrix*.

Tables 5, 6, 7, 8, 9 and 10 show the performance and the significance matrices for the static data, the step move data with $\Delta d = 5\%$, the step move data with $\Delta d = 10\%$, the shift data, the match data based on F_{Sph} and the match data based on F_{Ros} respectively. The average RMAE among the 30 runs of each algorithm and the ranks of the algorithms in regard to the average RMAE are shown in the tables. Table 5 shows that D-PSO is better than C-PSO in generating time-invariant models based on the static data Π_{Sph}^{stat}. C-PSO is better than D-GP which is better than C-GP, P-GA and PNN. A significant difference can be found between the results obtained by the PSO algorithms (C-PSO and D-PSO) and those obtained by the evolutionary algorithms (D-GP, C-GP, P-GA and PNN). However, there is no significant difference between the results obtained by C-PSO and D-PSO, even if D-PSO can obtain a smaller RMAE than that obtained by C-PSO. In regard to the other static data sets (Π_{Ros}^{stat}, Π_{Ras}^{stat}, Π_{Gri}^{stat}, Π_{Ack}^{stat}), both the PSO algorithms, C-PSO and D-PSO can obtain a smaller average RMAE than that obtained by the evolutionary algorithms, D-GP, C-GP, P-GA and PNN. Also, significant differences exist between the results obtained by the PSO algorithms (C-PSO and D-PSO) and those obtained by the evolutionary algorithms (D-GP, C-GP, P-GA and PNN). Also, similar results can be found in Tables 6, 7, 8, 9 and 10 where the PSO algorithms are significantly better than the evolutionary algorithms in generating models based on the time-varying data.

Therefore, it can be concluded that the PSO algorithms are significantly better than the evolutionary algorithms.

3.4 Population diversity

An investigation of population diversities of C-PSO, D-PSO, D-GP, C-GP, P-GA and PNN is presented in this section. Maintaining population diversity in population-based algorithms like evolutionary algorithms or PSO is a key to preventing premature convergence and stagnation in local optima [11, 16, 40]. Thus it is essential to study the population diversities of the six algorithms during the search. Various diversity measures, which involve calculations of distance between two individuals in genetic programming for the development of models, have been widely studied [4, 49]. These distance measures calculate the distances between two individuals which are in a tree based representation in genetic programming. They indicate the number of different nodes and different terminals between two individuals. In this paper, we measure the distance between two individuals by counting the number of different terms of the polynomials represented by the two individuals in the four algorithms. If the terms in both polynomials are all identical, the distance between the two polynomials is zero. The distance between the two polynomials is larger when the number of different terms in the two polynomials is larger. For example, f_1 and f_2 are two polynomials represented by:

$$f_1 = x_1 + x_2 + x_1 \cdot x_3 + x_4^{\,2} + x_1 \cdot x_3 \cdot x_5$$

and $$f_2 = x_1 + x_2 + x_1 \cdot x_5 + x_4 + x_1 \cdot x_3 \cdot x_5$$

Both f_1 and f_2 contain the three terms x_1, x_2 and $x_1 \cdot x_3 \cdot x_5$, and the terms $x_1 \cdot x_3$ and $x_4^{\,2}$ in f_1 and the terms $x_1 \cdot x_5$ and x_4 in f_2 are different. Therefore, the number of terms which are different in f_1 and f_2 are 2, and the distance between f_1 and f_2 is defined to be 2.

The diversity measure of the population at the *g*-th generation is defined by the total sum of distances of individuals which is denoted as:

$$\sigma_g = \sum_{i=1}^{N_P} \sum_{j=i+1}^{N_P} d\left(s_g(i), s_g(j)\right)$$

where $s_g(i)$ and $s_g(j)$ are the *i*-th and the *j*-th individuals in the population at the *g*-th generation, and *d* is the distance measure between the two individuals.

The diversities of the populations throughout the generations were recorded for the four algorithms. Figure 4 shows the diversity plots which indicate the diversities of the individuals in the algorithms in generating the models based on the step move data with $\Delta d = 5\%$. Figure 4a, 4b, 4c, 4d and 4e shows the diversities for static data which are generated based on the benchmark functions F_{Sph}, F_{Ros}, F_{Ras}, F_{Gri} and F_{Ack} respectively. The diversities of the populations throughout the generations were recorded for the six algorithms. The five figures indicate that the diversities along the generations of the D-PSO are slightly higher than those of the C-PSO which are much higher than those of the evolutionary algorithms, D-GP, C-GP, P-GA and PNN. For the rest of the data (static data, step move data with $\Delta d = 10\%$, shift data, match data based on F_{Sph}, match data based on F_{Ros}), similar findings indicate that the diversities of the two PSO algorithms are much larger than those of the evolutionary algorithms.

The findings indicate the reason why the PSO algorithms, D-PSO and C-PSO can obtain significantly better results than the evolutionary algorithms, D-GP, C-GP, P-GA and PNN. The diversities of the individuals of the PSO algorithms can be maintained along the search in both earlier and later generations, while the individuals of the evolutionary algorithms converged in the earlier generation. Therefore, the PSO

algorithms are more likely to explore the solution space, as the diversity of the individuals of the algorithms can be maintained. In regard to the effectiveness of the two PSO algorithms, since the diversities of the populations of C-PSO are only slightly smaller than those of the D-PSO, D-PSO can obtain only slightly better results than those obtained by C-PSO.

4 Conclusion and further work

In this paper, a particle swarm optimization algorithm has been proposed for developing polynomial models in which both time-invariant and time-varying characteristics are represented. The individuals of the PSO are represented by arithmetic operations and system variables, which are the components of the polynomial models. A set of dynamic benchmark functions whose optima or landscapes vary over time was employed to evaluate the performance of the PSO. The PSO algorithms, C-PSO and D-PSO, were used to generate models based on both the time-invariant and time-varying data sets. The evolutionary algorithms, D-GP, C-GP, P-GA and PNN, were also included in the experiments for comparison. Results show that the PSO algorithms significantly outperform the evolutionary algorithms in generating models based on both the time-invariant or time-varying data sets. It is observed that the evolutionary algorithms converge faster in the earlier generations when compared with the PSO algorithms. In contrast, the PSO algorithms can obtain better solutions in the later generations of the runs. The performance obtained in the results can be explained by the diversity measures and the fact that significant differences of diversities between the PSO algorithms and the evolutionary algorithms exist.

In future work, we will enhance the effectiveness of the PSO by the hybridization of the evolutionary algorithm and the PSO algorithm. Here the evolutionary algorithm will be implemented to localize the potential solutions in the early generations and the PSO algorithm will be implemented in order to continue to explore the solution space to avoid pre-mature convergence in late generations. The resulting algorithm will be further validated by solving real-time traffic flow forecasting problems, which are time varying in nature.

References

[1] F.J. Berlanga, A.J. Rivera, M.L. del Jesus and F. Herrera, GP-COACH: Genetic programming-based learning of COmpact and ACcurate fuzzy rule-based classification systems for high-dimensional problems, Information Sciences, vol. 15, no. 8, pp. 1183-1200, 2010.

[2] T. Blackwell and J. Branke, Multiswarms, exclusion, and anti-convergence in dynamic environments, IEEE Transactions on Evolutionary Computation, vol. 10, no. 4, pp. 459-472, 2006.

[3] S. Billings, M. Korenberg and S. Chen, "Identification of nonlinear outputaffine systems using an orthogonal least-squares algorithm", International Journal of Systems Science, vol. 19, 1559-1568, 1988.

[4] E.K. Burke, S. Gustafson and G. Kendall, Diversity in genetic programming: an analysis of measures and correlation with fitness, IEEE Transactions on Evolutionary Computation, vol. 8, no. 1, pp. 47-62, 2004.

[5] Y.C. Chen, T.H.S. Li and Y.C. Yeh, EP-based kinematic control and adaptive fuzzy sliding mode dynamic control for wheeled mobile robots, Information Sciences, vol. 179, no. 1, pp. 180-195, 2009.

[6] G. Chen and C.P. Low, Coordinating agents in shop floor environments from a dynamic systems perspective, IEEE Transactions on Industrial Informatics, vol. 2, no. 4, pp. 269-280, 2006.

[7] S. Chen, S. Billings and W. Luo, Orthogonal least squares methods and their application to non-linear system identification, International Journal of Control, vol. 50, pp. 1873-1896, 1988.

[8] Y.P. Chen, W.C. Peng and M.C. Jian, Particle swarm optimization with recombination and dynamic linkage discovery, IEEE Transactions on Systems, Man

and Cybernetics - Part B, vol. 37, no. 6, pp. 1460-1470, 2007.

[9] Y. Chen, B. Yang, Q. Meng and A. Abraham, Time series forecasting using a system of ordinary differential equations, Information Sciences, vol. 181, no. 1, pp. 106-114, 2011.

[10] M. Clerc and J. Kennedy, The particle swarm – explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation, vol. 6, no. 1, pp. 58-73, 2002.

[11] P. Day and A.K. Nandi, Robust text-independent speaker verification using genetic programming, IEEE Transactions on Audio, Speech and Language Processing, vol. 15, no. 1, pp. 285-295, 2007.

[12] R. Eberhart and J. Kennedy, A new optimizer using particle swarm theory, in Proceedings of the 6th International Symposium on Micro Machine and Human Science, IEEE Service Center, Nagoya, Oct. 1995, pp.39-43.

[13] R.C. Eberhart and Y. Shi, Comparison between genetic algorithms and particle swarm optimization, in Evolutionary Programming VII. New York: Springer-Verlag, vol. 1447, Lecture Notes in Computer Science, 1998, pp. 611-616.

[14] R.C. Eberhart, and Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in Proceedings of the Congress on Evolutionary Computing, IEEE Service Center, vol. 1, Jul. 2000, pp.84-88.

[15] R.C. Eberhart and Y. Shi, Particle swarm optimization: developments, applications and resources, in Proc. Congress on Evolutionary Computation (Hawaii), 2001, pp 81–6

[16] K.B. Edmund, G. Steven and K. Graham, Diversity in genetic programming: an analysis of measures and correlation with fitness, IEEE Transactions on Evolutionary Computation, vol. 8, no. 1, pp. 47-62, 2004.

[17] A. Ekart and S.Z. Nemeth, A metric for genetic programs and fitness sharing, in Proceedings of the European Conference on Genetic Programming, vol. 1802, pp. 259-270, 2000.

[18] D. Gabor, W. Wides and R. Woodcock, A universal nonlinear filter predictor and simulator which optimizes itself by a learning process, Proceedings of IEE, vol. 108-B, pp. 422-438, 1961.

[19] S. Garcia, D. Molina, M. Lozano and F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC' 2005 special session on real parameter optimization, Journal of Heuristics, vol. 15, pp. 617-644, 2009.

[20] D.E. Goldberg, Genetic algorithms in search, optimization and machine learning, United States of America: Addison Wesley Longman, Inc., 1989.

[21] H. Iba, Inference of differential equation models by genetic programming, Information Sciences, vol. 178, pp. 4453-4468, 2008.

[22] A. Izadian, P. Khayyer and P. Famouri, Fault diagnosis of time-varying parameter systems with application in MEMS LCRs, IEEE Transactions on Industrial Electronics, vol. 56, no. 4, pp. 973-978, 2009.

[23] S. Janson and M. Middendorf, A hierarchical particle swarm optimizer for noisy and dynamic environments, Genetic Programming and Evolvable Machines, vol. 7, pp. 329-354, 2006.

[24] Y. Jin and B. Sendhoff, Constructing dynamic optimization test problems using the multi-objective optimization concept, In G. R. Raidl, editor, Applications of evolutionary computing, Lecture Notes on Computer Sciences 3005, pp. 525-536. Springer, 2004.

[25] J. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Evolution, MIT Press: Cambridge, 1992.

[26] J. Koza, Genetic Programming II: automatic discovery of reusable programs, MIT Press, 1994.

[27] J.R. Koza, M.J. Streeter and M.A. Keane, Routine high-return human-competitive automated problem-solving by means of genetic programming, Information Sciences, vol. 178, no. 23, pp. 4434-4452, 2008.

[28] J. Kennedy and R.C. Eberhart, Particle Swarm Optimization, in IEEE International Conference on Neural Networks, 1995, pp. 1942-1948.

[29] J. Kennedy and R.C. Eberhart, Swarm Intelligence. Morgan Kaufmann Publishers, 2001.

[30] J. Kennedy and R.C. Eberhart, A discrete binary version of the particle swarm algorithm, IEEE Proceedings of International Conference on Systems, Man and Cybernetics, vol. 5, pp. 4104-4108, 1997.

[31] C.J. Liao, C.T. Tseng and P. Luran, A discrete version of particle swarm optimization for flowshop scheduling problems, Computer and Operations Research, vol. 34, pp. 3099-3111, 2007.

[32] C. Li and S. Yang, A generalized approach to construct benchmark problems for dynamic optimization, Proceedings of the 7th International Conference on Simulated Evolution and Learning, pp. 391-400, 2008.

[33] C. Li, S. Yang, T.T. Nguyen, E.L. Yu, X. Yao, Y. Jin, H.G. Beyer and P.N. Suganthan, Benchmark Generator for CEC 2009 Competition on Dynamic

Optimization, Technical report, University of Leicester and University of Birmingham, UK, 2009.

[34] D.R. Liu, C.H. Lai and W.J. Lee, A hybrid of sequential rules and collaborative filtering for product recommendation, Information Sciences, vol. 179, no. 20, pp. 3505-3519, 2009.

[35] G. Li, J.F. Wang, K.H. Lee and K.S. Leung, Instruction-matrix-based genetic programming, IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics, vol. 38, no. 4, pp. 1036-1049, 2008.

[36] T. Li, Lei Guo and X. Lin, Improved delay-dependent bounded real lemma for uncertain time-delay systems, Information Sciences, vol. 179, no. 20, pp. 3711-3729, 2009.

[37] Y. Lin and B. Bhanu, Object detection via feature synthesis using MDL-based genetic programming, IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics, vol. 35, no. 3, pp. 538-547, 2005.

[38] J. Madar, J. Abonyi and F. Szeifert, Genetic programming for the identification of nonlinear input output models, Industrial and Engineering Chemistry Research, vol. 44, pp. 3178 – 3186, 2005.

[39] B. McKay, M.J. Willis and G.W. Barton, Steady-state modeling of chemical processes using genetic programming, Computers and Chemical Engineering, vol. 21, no. 9, pp. 981-996, 1997.

[40] R.I. McKay, Fitness sharing genetic programming, Proceedings of the Genetic and Evolutionary Computation Conference, pp. 435-442, 2000.

[41] N. Motoi, M. Ikebe and K. Ohnishi, Real-time gait planning for pushing motion of humanoid robot, IEEE Transactions on Industrial Informatics, vol. 3, no. 2, pp. 154-163, 2007.

[42] R.W. Morrison and K.A. De Jong, A test problem generator for non-stationary environments, Proceedings of the IEEE Congress on Evolutionary Computation, pp. 2047-2053, 1999.

[43] D.P. Muni, N.R. Pal and J. Das, Genetic programming for simultaneous feature selection and classifier design, IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics, vol. 36, no. 1, pp. 106-117, 2006.

[44] K. Natori, R. Oboe and K. Ohnishi, Stability analysis and practical design procedure of time delayed control systems with communication disturbance observer, IEEE Transactions on Industrial Informatics, vol. 4, no. 3, pp. 185-197, 2007.

[45] B. Naudts and L. Kallel, A comparison of predictive measures of problem difficulty in evolutionary algorithms, IEEE Transactions on Evolutionary Computation, vol. 4, no. 1, 2000.

[46] M.O. Neill and Conor Ryan, Grammatical Evolution, IEEE Transactions on Evolutionary Computation, vol. 5, no. 4, pp. 349-358, 2001.

[47] M.O. Neill and A. Brabazon, Grammatical Swarm, Genetic and Evolutionary Computation Conference, vol. 1, pp. 163-174, 2004.

[48] M.O. Neill and A. Brabazon, Grammatical Swarm: The generation of programs by social programming, Natural Computing, vol. 5, pp. 443-462, 2006.

[49] X.H. Nguyen, R.I. McKay, D. Essam and H.A. Abbass, Toward an alternative comparison between different genetic programming systems, Proceedings of European Conference on Genetic Programming, pp. 67-77, 2004.

[50] S.K. Oh and W. Pedrycz, Multi-layer self-organizing polynomial neural networks and their development with the use of genetic algorithms, Journal of the Franklin Institute, vol. 343, pp. 125-136, 2006.

[51] Q.K. Pan, M.F. Tasgetiren and Y.C. Liang, A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem, Computers and Operations Research, vol. 35, pp. 2907-2839, 2008.

[52] D. Parrott and X. Li, Locating and tracking multiple dynamic optima by a particle swarm model using speciation, IEEE Transactions on Evolutionary Computation, vol. 10, no. 4, pp. 440-458, 2006.

[53] K.E. Parsopoulos and M.N. Vrahatis, On the computation of all global minimizers through particle swarm optimization, IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, pp. 211-224, 2004.

[54] G. Potgieter and A.P. Engelbrecht, Genetic algorithms for the structural optimization of learned polynomial expression", Applied Mathematics and Computation, vol. 186, pp. 1441-1466, 2007.

[55] C.R. Reeves, Genetic algorithms and neighbourhood search, Evolutionary Computing: AISB Workshop, pp. 115-130, 1994.

[56] K. Rodriguez-Vazquez, C.M. Fonseca and P.J. Fleming, Identifying the structure of nonlinear dynamic systems using multiobjective genetic programming, IEEE Transactions on Systems, Man and Cybernetics – Part A, vol. 34, no. 4, pp. 531-545, 2004.

[57] T. Shibata and T. Murakami, Null space motion control by PID control considering passivity in redundant manipulator, IEEE Transactions on Industrial Informatics, vol. 4, no. 4, pp. 261-270, 2008.

[58] C.T. Tseng and C.J. Liao, A discrete particle swarm optimization for lot-streaming flowshop scheduling problem, European Journal of Operational Research, vol. 191, pp. 360-373, 2008.

[59] R.J.M. Vaessens, E.H.L. Aarts, and J.K. Lenstra, A local search template, in Proc Parallel Problem-Solving from Nature 2, 1992, pp. 65-74.

[60] N. Wagner, Z. Michalewicz, M. Khouja and R.R. McGregor, Time series forecasting for dynamic environments: the DyFor genetic program model, IEEE Transactions on Evolutionary Computation, vol. 11, no. 4, pp. 433-452, 2007.

[61] J. Wang, Z. Fan, J.P. Terpenny and E.D. Goodman, Knowledge interaction with genetic programming in mechatronic systems design using bond graphs, IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and Reviews, vol. 35, no. 2, pp. 172-182, 2005.

[62] Y. Wang and Y. Yang, Particle swarm optimization with preference order ranking for multi-objective optimization, Information Sciences, vol. 179, no. 12, pp. 1944-1959, 2009.

[63] Y.G. Woldesenbet and G.G. Yen, Dynamic evolutionary algorithm with variable relocation, IEEE Transactions on Evolutionary Computation, vol. 13, no. 3, pp. 500-513, 2009.

[64] S.X. Yang and X. Yao, Population-based incremental learning with associative memory for dynamic environments, IEEE Transactions on Evolutionary Computation, vol. 12, no. 5, pp. 542-561, 2008.

[65] A. Zafra and S. Ventura, G3P-MI: a genetic programming algorithm for multiple instance learning, Information Sciences, vol. 180, no. 23, pp. 4496-4513, 2010.

[66] Y. Zhang, K.M. Jan, K.H. Ju and K.H. Chon, Representation of time-varying nonlinear systems with time-varying principle dynamic modes, IEEE Transactions on Biomedical Engineering, vol. 54, no. 11, pp. 1983-1992, 2007.

Appendix

{
Step 1: g=0
Step 2: Initialize $\Omega(g)=[\theta_1(g), \theta_2(g),\ldots \theta_{POP}(g)]$
// $\Omega(g)$ is the population of the g-th generation.
// $\theta_i(g)$ is the i-th individual of $\Omega(g)$.
//where $\theta_k(g) = \sum x_i + \sum x_i\, x_j + \ldots..\prod x_i$
Step 3: **Assign system coefficients a(t) in all** $\theta_k(g)$ **by LSM**
//where $\theta_i(g)' = a_0(t) + \sum a_i(t)\, x_i + \sum a_{ij}(t)\, x_i\, x_j +$
// $\ldots..+ a_{12..Nterm}(t) \prod x_i$
Step 4: Evaluate all $\theta_k(g)'$ based on (3)
while (Terminational condition not fulfilled) **do {**
Step 5: Parent Selection $\Omega(g+1) =[\theta_1(g+1), \theta_2(g+1),$
$\ldots\theta_{POP}(g+1)]$
// where $\theta_k(g+1) = \sum x_i + \sum x_i\, x_j +\ldots\prod x_i$
Step 6: Crossover $\Omega(g+1)$
Step 7: Mutation $\Omega(g+1)$
Step 8: **Assign parameters a(k) in all** $\theta_k(g+1)$ **by LSM**
//where $\theta_k(g+1)' = a_0(t) + \sum a_i(t)\, x_i + \sum a_{ij}(t)\, x_i\, x_j +$
// $\ldots..+ a_{12..Nterm}(t) \prod x_i$
Step 9: Evaluate all $\theta_k(g+1)'$ based on (3)
Step 10: $\Omega(g)= \Omega(g+1)$
Step 11: $g=g+1$
}

Figure 2: The pseudocode of the genetic programming GP

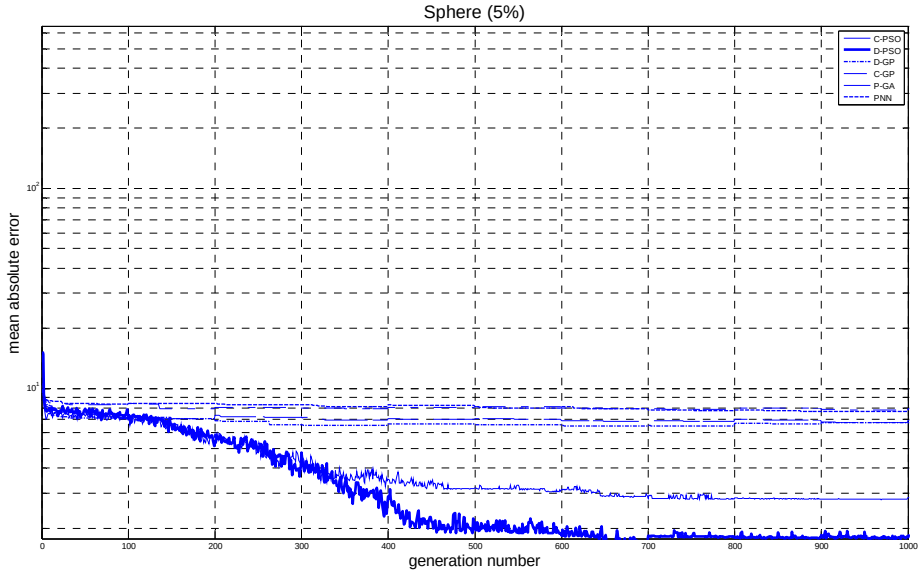

Figure 3a: Convergence plot for step moving data Π_{Sph}^{Step-5} (Sphere function with $\Delta d = 5\%$)

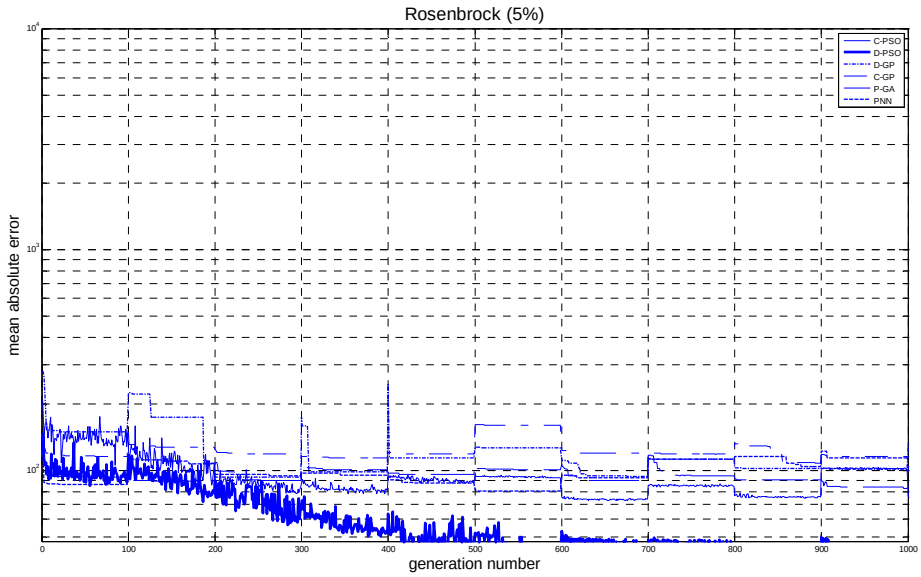

Figure 3b: Convergence plot for step moving data Π_{Ros}^{Step-5} (Rosenbrock function with $\Delta d = 5\%$)

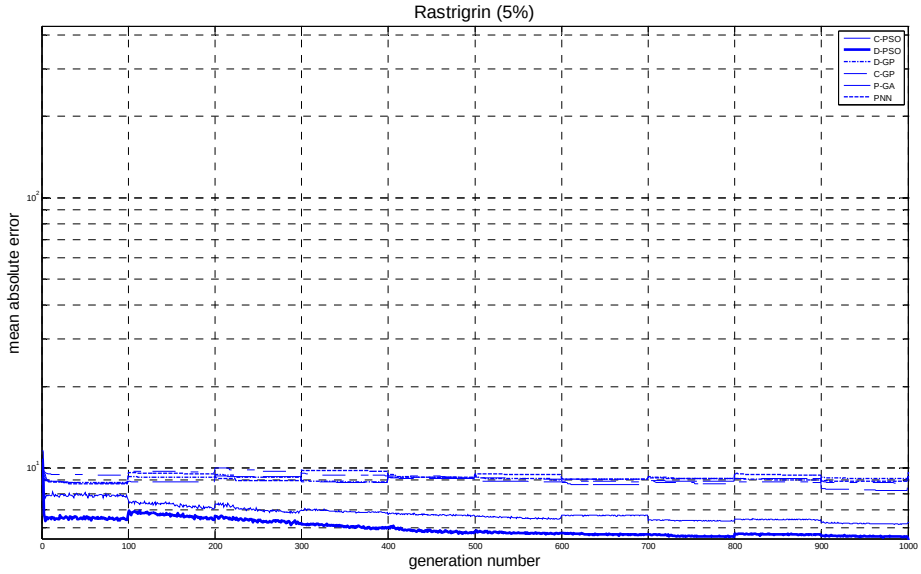

Figure 3c: Convergence plot for step moving data Π_{Ras}^{Step-5} (Rastrigrin function with $\Delta d = 5\%$)

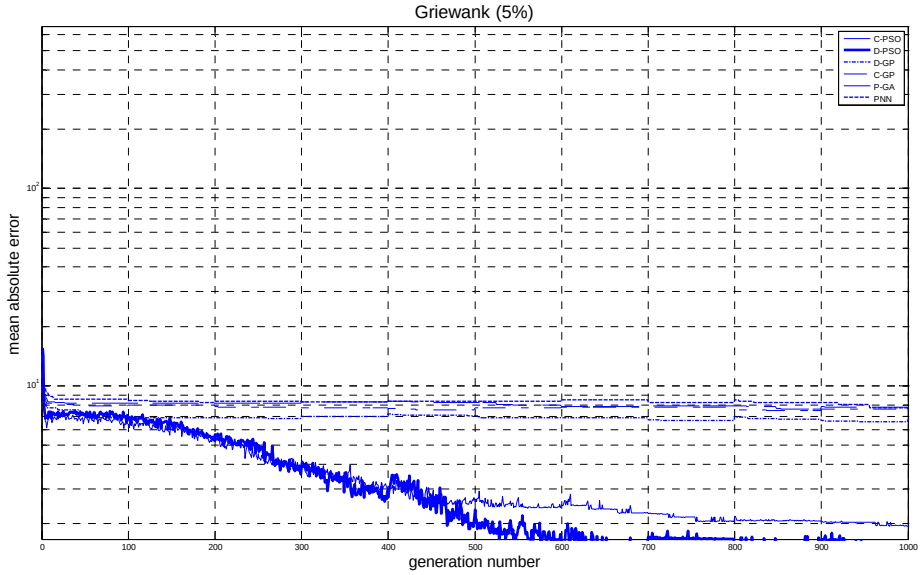

Figure 3d: Convergence plot for step moving data Π_{Gri}^{Step-5} (Griewank function with $\Delta d = 5\%$)

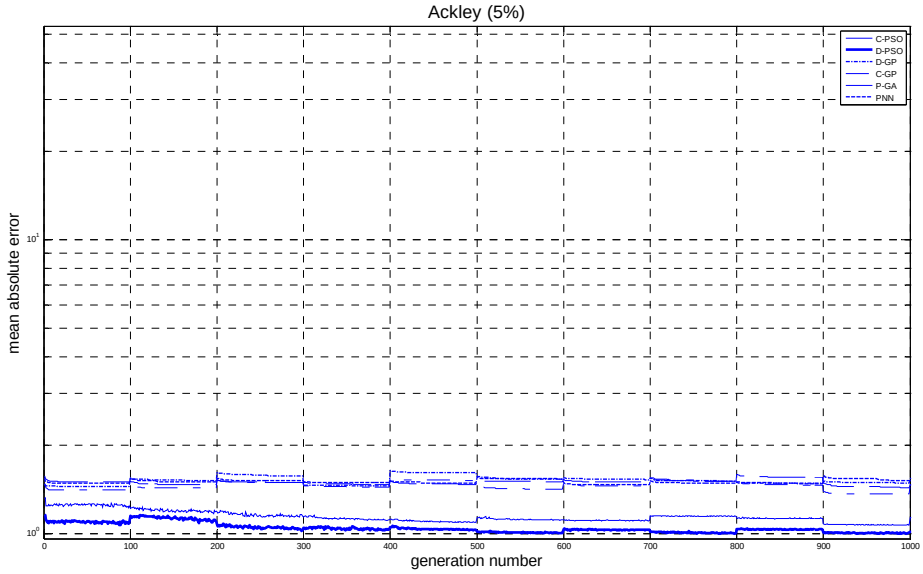

Figure 3e: Convergence plot for step moving data Π_{Ackley}^{Step-5} (Ackley function with $\Delta d = 5\%$)

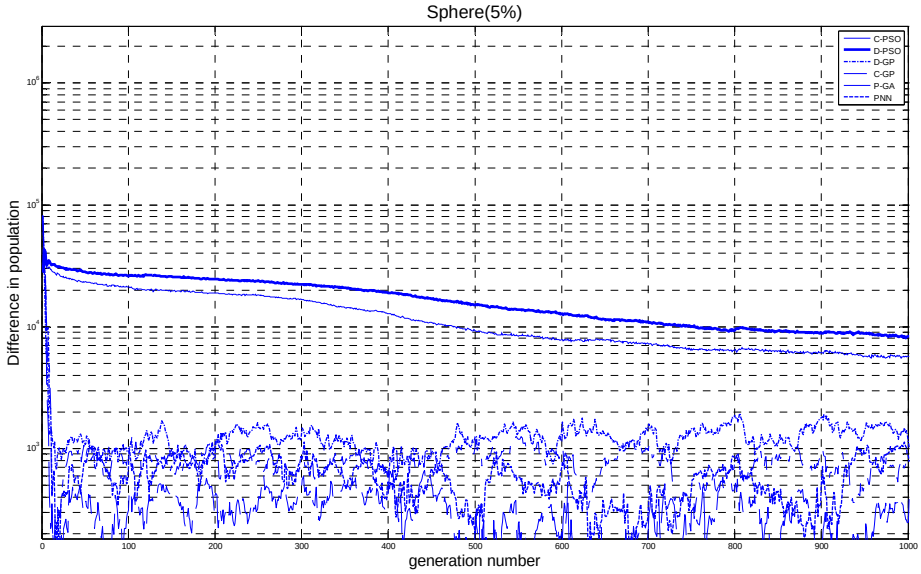

Figure 4a: Diversity plot for step moving data Π_{Sph}^{Step-5} (Sphere function with $\Delta d = 5\%$)

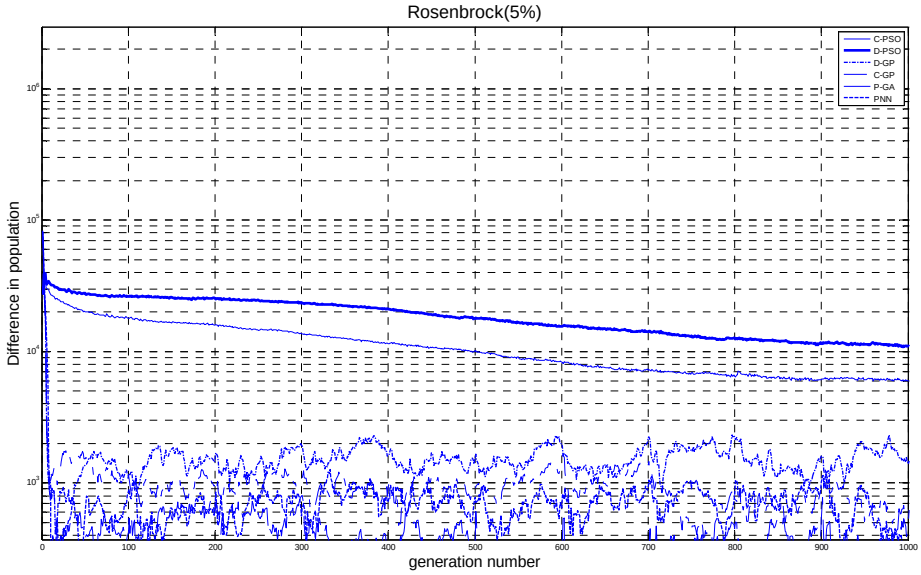

Figure 4b: Diversity plot for step moving data Π_{Ros}^{Step-5} (Rosenbrock function with $\Delta d = 5\%$)

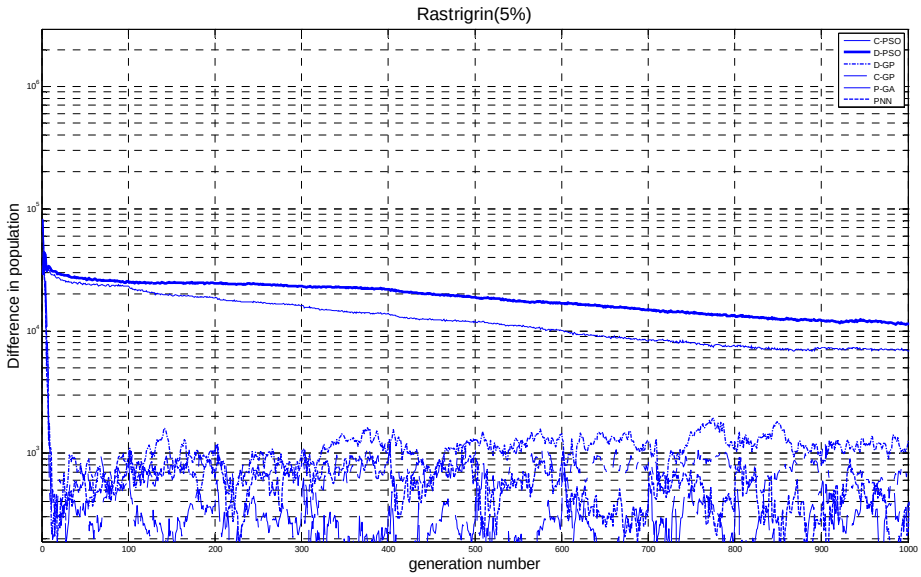

Figure 4c: Diversity plot for step moving data Π_{Ras}^{Step-5} (Rastrigrin function with $\Delta d = 5\%$)

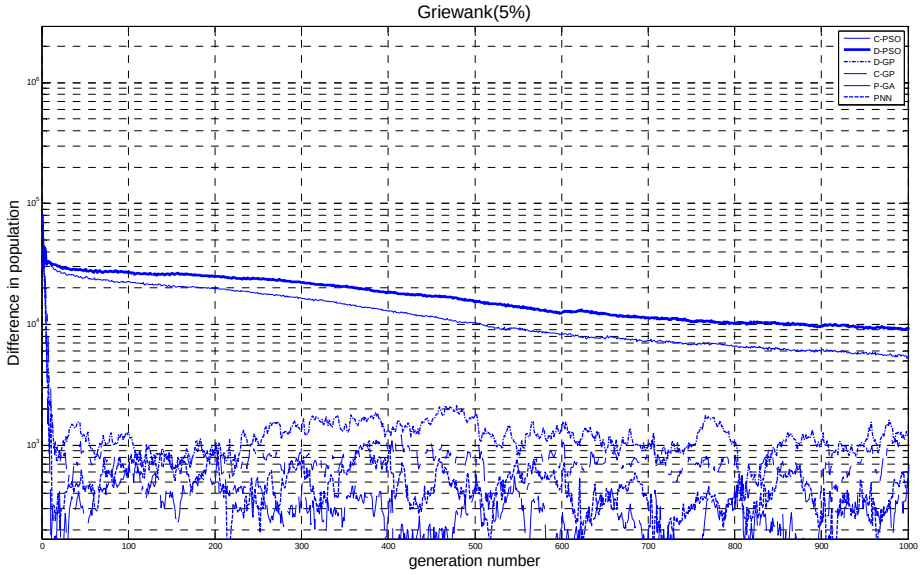

Figure 4d: Diversity plot for step moving data Π_{Gri}^{Step-5} (Griewank function with $\Delta d = 5\%$)

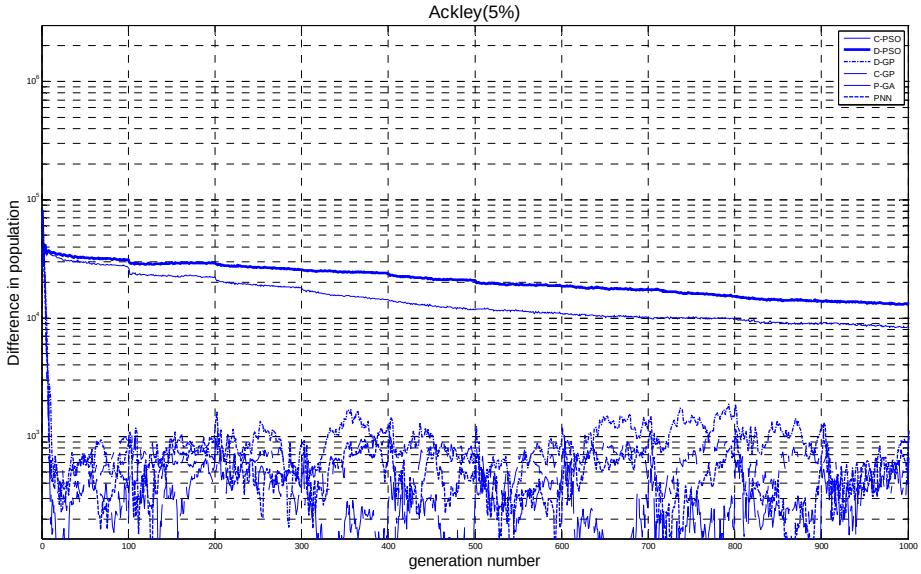

Figure 4e: Diversity plot for step moving data Π_{Ackley}^{Step-5} (Ackley function with $\Delta d = 5\%$)

Table 5: Average RMAE and ranks for the static data (Π^{stat}_{Sph}, Π^{stat}_{Ros}, Π^{stat}_{Ras}, Π^{stat}_{Gri}, Π^{stat}_{Ack})

Algorithm	Static F_{Sph} (Π^{stat}_{Sph})							Static F_{Ros} (Π^{stat}_{Ros})							Static F_{Ras} (Π^{stat}_{Ras})							Static F_{Gri} (Π^{stat}_{Gri})							Static F_{Ack} (Π^{stat}_{Ack})							Mean rA
	oP (rA)	S-matrix						oP (rA)	S-matrix						oP (rA)	S-matrix						oP (rA)	S-matrix						oP (rA)	S-matrix						
		1	2	3	4	5	6		1	2	3	4	5	6		1	2	3	4	5	6		1	2	3	4	5	6		1	2	3	4	5	6	
1 C-PSO	0.8873 (1)	_	_	X	X	X	X	1.9252 (2)	_	_	X	X	X	X	3.6538 (2)	_	_	X	X	X	X	2.9008 (2)	_	_	X	X	X	X	0.8178 (2)	_	X	X	X	X	X	1.8
2 D-PSO	1.3661 (2)	_	_	X	X	X	X	1.0401 (1)	_	_	X	X	X	X	2.2758 (1)	_	_	X	X	X	X	1.8290 (1)	_	_	X	X	X	X	0.7747 (1)	X	_	X	X	X	X	1.2
3 D-GP	4.3595 (3)	X	X	_	X	_	_	4.2012 (3)	X	X	_	_	_	X	5.6365 (3)	X	X	_	_	_	X	3.4597 (3)	X	X	_	_	_	_	0.9546 (3)	X	X	_	X	X	X	3
4 C-GP	5.7800 (6)	X	X	X	_	_	_	5.2979 (5)	X	X	_	_	_	_	6.0036 (4)	X	X	_	_	_	_	5.8344 (6)	X	X	_	_	_	_	1.0398 (5)	X	X	X	_	_	_	5.2
5 P-GA	4.7552 (5)	X	X	_	_	_	_	5.1909 (4)	X	X	_	_	_	_	6.0764 (5)	X	X	_	_	_	_	5.1195 (5)	X	X	_	_	_	_	1.0069 (4)	X	X	X	_	_	_	4.6
6 PNN	4.4885 (4)	X	X	_	_	_	_	5.8052 (6)	X	X	X	_	_	_	6.3742 (6)	X	X	X	_	_	_	4.9114 (4)	X	X	_	_	_	_	1.1207 (6)	X	X	X	_	_	_	5.2

oP – Average RMAE obtained by the algorithms; rA – rank; ‘X’ – difference between the two algorithms is significant; ‘_’ - difference between the two algorithms is not significant;

Table 6: Average RMAE and ranks for the step move data with $\Delta d = 5\%$ (Π_{Sph}^{Step-5}, Π_{Ros}^{Step-5}, Π_{Ras}^{Step-5}, Π_{Gri}^{Step-5}, Π_{Ack}^{Step-5})

Algorithm	Step F_{Sph} - $\Delta d = 5\%$ (Π_{Sph}^{Step-5})							Step F_{Ros} - $\Delta d = 5\%$ (Π_{Ros}^{Step-5})							Step F_{Ras} - $\Delta d = 5\%$ (Π_{Ras}^{Step-5})							Step F_{Gri} - $\Delta d = 5\%$ (Π_{Gri}^{Step-5})							Step F_{Ack} - $\Delta d = 5\%$ (Π_{Ack}^{Step-5})							Mean rA
	oP (rA)	S-matrix						oP (rA)	S-matrix						oP (rA)	S-matrix						oP (rA)	S-matrix						oP (rA)	S-matrix					
		1	2	3	4	5	6		1	2	3	4	5	6		1	2	3	4	5	6		1	2	3	4	5	6		1	2	3	4	5	6	
1 C-PSO	2.8118 (2)	_	_	X	X	X	X	2.0365 (2)	_	_	X	X	X	X	4.9994 (2)	_	X	X	X	X	X	4.4084 $\times 10^1$ (2)	_	X	X	X	X	X	0.8790 (2)	_	_	X	X	X	X	2
2 D-PSO	1.7906 (1)	_	_	X	X	X	X	1.6411 (1)	_	_	X	X	X	X	4.9159 (1)	X	_	X	X	X	X	2.6067 $\times 10^1$ (1)	X	_	X	X	X	X	0.7767 (1)	_	_	X	X	X	X	1
3 D-GP	4.1972 (3)	X	X	_	X	_	_	4.8945 (3)	X	X	_	_	_	_	6.6285 (5)	X	X	_	_	_	_	6.0471 $\times 10^1$ (3)	X	X	_	_	_	_	2.7233 (4)	X	X	_	_	_	_	3.6
4 C-GP	6.2698 (6)	X	X	X	_	_	_	6.0860 (6)	X	X	_	_	_	_	6.7806 (6)	X	X	_	_	_	_	8.3000 $\times 10^1$ (6)	X	X	_	_	_	_	3.0748 (6)	X	X	_	_	_	_	6
5 P-GA	5.9230 (5)	X	X	_	_	_	_	5.2762 (4)	X	X	_	_	_	_	6.2917 (3)	X	X	_	_	_	_	6.3092 $\times 10^1$ (4)	X	X	_	_	_	_	2.9372 (5)	X	X	_	_	_	_	4.2
6 PNN	5.1709 (4)	X	X	_	_	_	_	5.7928 (5)	X	X	_	_	_	_	6.5860 (4)	X	X	_	_	_	_	6.7838 $\times 10^1$ (5)	X	X	_	_	_	_	2.7220 (3)	X	X	_	_	_	_	4.2

oP – Average RMAE obtained by the algorithms; rA – rank; 'X' – difference between the two algorithms is significant; '_' - difference between the two algorithms is not significant;

Table 7: Average RMAE and ranks for the step move data sets with $\Delta d = 10\%$ ($\Pi_{Sph}^{Step-10}$, $\Pi_{Ros}^{Step-10}$, $\Pi_{Ras}^{Step-10}$, $\Pi_{Gri}^{Step-10}$, $\Pi_{Ack}^{Step-10}$)

Alg.	Step F_{Sph} - $\Delta d = 10\%$ ($\Pi_{Sph}^{Step-10}$)							Step F_{Ros} - $\Delta d = 10\%$ ($\Pi_{Sph}^{Step-10}$)							Step F_{Ras} - $\Delta d = 10\%$ ($\Pi_{Sph}^{Step-10}$)							Step F_{Gri} - $\Delta d = 10\%$ ($\Pi_{Sph}^{Step-10}$)							Step F_{Ack} - $\Delta d = 10\%$ ($\Pi_{Sph}^{Step-10}$)							Mean rA
	oP (rA)	S-matrix						oP (rA)	S-matrix						oP (rA)	S-matrix						oP (rA)	S-matrix						oP (rA)	S-matrix					
		1	2	3	4	5	6		1	2	3	4	5	6		1	2	3	4	5	6		1	2	3	4	5	6		1	2	3	4	5	6	
1 C-PSO	3.8595 (2)	_	_	X	X	X	X	4.1047 (2)	_	X	X	X	X	X	5.6531 (2)	_	_	X	X	X	X	6.7125×10^1 (2)	_	X	X	X	X	X	6.1591×10^1 (2)	_	_	X	X	X	X	2
2 D-PSO	3.6369 (1)	_	_	X	X	X	X	3.1930 (1)	X	_	X	X	X	X	5.6379 (1)	_	_	X	X	X	X	5.1045×10^1 (1)	X	_	X	X	X	X	3.5942×10^1 (1)	_	_	X	X	X	X	1
3 D-GP	5.1225 (5)	X	X	_	X	_	_	7.2864 (5)	X	X	_	_	_	_	7.2221 (4)	X	X	_	_	_	_	0.8528×10^2 (3)	X	X	_	X	_	_	0.7737×10^2 (3)	X	X	_	_	_	_	4
4 C-GP	7.1423 (6)	X	X	X	_	_	_	7.5441 (6)	X	X	_	_	_	_	7.5313 (5)	X	X	_	_	_	_	1.0793×10^2 (6)	X	X	X	_	_	_	1.0155×10^2 (6)	X	X	_	_	_	_	5.8
5 P-GA	5.9374 (4)	X	X	_	_	_	_	7.0326 (3)	X	X	_	_	_	_	7.1957 (3)	X	X	_	_	_	_	1.0019×10^2 (5)	X	X	_	_	_	_	0.9216×10^2 (5)	X	X	_	_	_	_	4
6 PNN	5.3002 (3)	X	X	_	_	_	_	7.1971 (4)	X	X	_	_	_	_	7.9044 (6)	X	X	_	_	_	_	0.8770×10^2 (4)	X	X	_	_	_	_	0.8781×10^2 (4)	X	X	_	_	_	_	4.2

oP – Average RMAE obtained by the algorithms; rA – rank; 'X' – difference between the two algorithms is significant; '_' - difference between the two algorithms is not significant;

Table 8: Average RMAE and ranks for the shift data (Π_{Sph}^{shift}, Π_{Ros}^{shift}, Π_{Ras}^{shift}, Π_{Gri}^{shift}, Π_{Ack}^{shift})

Alg.	Shift F_{Sph} (Π_{Sph}^{shift})							Shift F_{Ros} (Π_{Ros}^{shift})							Shift F_{Ras} (Π_{Ras}^{shift})							Shift F_{Gri} (Π_{Gri}^{shift})							Shift F_{Ack} (Π_{Ack}^{shift})							Mean rA
	oP	S-matrix						oP	S-matrix						oP	S-matrix						oP	S-matrix						oP	S-matrix						
	(rA)	1	2	3	4	5	6	(rA)	1	2	3	4	5	6	(rA)	1	2	3	4	5	6	(rA)	1	2	3	4	5	6	(rA)	1	2	3	4	5	6	
1 C-PSO	8.01×10^{-3} (1)	_	_	X	X	X	X	6.90×10^{-5} (2)	_	X	X	X	X	X	0.96 (2)	_	_	X	X	X	X	2.46×10^{-1} (2)	_	X	X	X	X	X	3.56×10^{-1} (2)	_	_	X	X	X	X	1.8
2 D-PSO	8.55×10^{-3} (2)	_	_	X	X	X	X	3.60×10^{-5} (1)	X	_	X	X	X	X	0.90 (1)	_	_	X	X	X	X	0.48×10^{-1} (1)	X	_	X	X	X	X	3.27×10^{-1} (1)	_	_	X	X	X	X	1.2
3 D-GP	15.32×10^{-3} (6)	X	X	_	_	_	_	7.45×10^{-5} (4)	X	X	_	_	_	_	2.11 (3)	X	X	_	_	_	_	2.73×10^{-1} (3)	X	X	_	X	_	_	3.66×10^{-1} (3)	X	X	_	_	X	_	3.8
4 C-GP	12.98×10^{-3} (3)	X	X	_	_	_	_	7.58×10^{-5} (5)	X	X	_	_	_	_	3.09 (6)	X	X	_	_	_	_	3.71×10^{-1} (6)	X	X	X	_	_	_	3.74×10^{-1} (4)	X	X	_	_	_	_	4.8
5 P-GA	13.26×10^{-3} (4)	X	X	_	_	_	_	7.20×10^{-5} (3)	X	X	_	_	_	_	2.85 (5)	X	X	_	_	_	_	3.47×10^{-1} (5)	X	X	_	_	_	_	4.57×10^{-1} (6)	X	X	X	_	_	_	4.6
6 PNN	14.31×10^{-3} (5)	X	X	_	_	_	_	7.83×10^{-5} (6)	X	X	_	_	_	_	2.56 (4)	X	X	_	_	_	_	3.33×10^{-1} (4)	X	X	_	_	_	_	3.85×10^{-1} (5)	X	X	_	_	_	_	4.8

oP – Average RMAE obtained by the algorithms; rA – rank; 'X' – difference between the two algorithms is significant; '_' - difference between the two algorithms is not significant;

Table 9: Average RMAE and ranks for the match data based on F_{Sph} ($\Pi^{match}_{Sph-Ros}$, $\Pi^{match}_{Sph-Ras}$, $\Pi^{match}_{Sph-Gri}$, $\Pi^{match}_{Sph-Ack}$)

Alg.	Match F_{Sph} - F_{Ros} ($\Pi^{match}_{Sph-Ros}$)							Match F_{Sph} - F_{Ras} ($\Pi^{match}_{Sph-Ras}$)							Match F_{Sph} - F_{Gri} ($\Pi^{match}_{Sph-Gri}$)							Match F_{Sph} - F_{Ack} ($\Pi^{match}_{Sph-Ack}$)							Mean rA
	oP (rA)	S-matrix						oP (rA)	S-matrix						oP (rA)	S-matrix						oP (rA)	S-matrix						
		1	2	3	4	5	6		1	2	3	4	5	6		1	2	3	4	5	6		1	2	3	4	5	6	
1 C-PSO	2.3821×10^{-4} (2)	_	_	X	X	X	X	3.0658 (2)	_	X	X	X	X	X	2.9197 (2)	_	_	X	X	X	X	1.0771×10^{-1} (2)	_	X	X	X	X	X	2
2 D-PSO	1.9713×10^{-4} (1)	_	_	X	X	X	X	0.7883 (1)	X	_	X	X	X	X	2.7702 (1)	_	_	X	X	X	X	3.7706×10^{-2} (1)	X	_	X	X	X	X	1
3 D-GP	3.2735×10^{-3} (3)	X	X	_	_	_	_	4.5844 (3)	X	X	_	_	_	_	5.0644 (3)	X	X	_	X	_	_	2.0051×10^{-1} (3)	X	X	_	X	X	X	3
4 C-GP	4.1303×10^{-4} (5)	X	X	_	_	_	_	4.6479 (4)	X	X	_	_	_	_	6.0078 (6)	X	X	X	_	_	_	2.9128×10^{-1} (6)	X	X	X	X	_	_	5.25
5 P-GA	4.0901×10^{-4} (4)	X	X	_	_	_	_	5.2263 (6)	X	X	_	_	_	_	5.7057 (5)	X	X	_	_	_	_	2.8324×10^{-1} (5)	X	X	X	_	_	_	5
6 PNN	4.4515×10^{-4} (6)	X	X	_	_	_	_	4.7533 (5)	X	X	_	_	_	_	5.6600 (4)	X	X	_	_	_	_	2.8057×10^{-1} (4)	X	X	X	_	_	_	4.75

oP – Average RMAE obtained by the algorithms; rA – rank; 'X' – difference between the two algorithms is significant; '_' - difference between the two algorithms is not significant;

Table 10: Average RMAE and ranks for the match data based on F_{Ros} ($\Pi^{match}_{Ros-Ras}$, $\Pi^{match}_{Ros-Gri}$, $\Pi^{match}_{Ros-Ack}$)

Alg.	Match F_{Sph} - F_{Ros} ($\Pi^{match}_{Sph-Ros}$)							Match F_{Sph} - F_{Ras} ($\Pi^{match}_{Sph-Ras}$)							Match F_{Sph} - F_{Gri} ($\Pi^{match}_{Sph-Gri}$)							Mean rA
	oP	S-matrix						oP	S-matrix						oP	S-matrix						
	(rA)	1	2	3	4	5	6	(rA)	1	2	3	4	5	6	(rA)	1	2	3	4	5	6	
1 C-PSO	2.1106 (2)	_	X	X	X	X	X	4.3659 (2)	_	X	X	X	X	X	1.2022×10^{-1} (2)	_	X	X	X	X	X	2
2 D-PSO	1.5180 (1)	X	_	X	X	X	X	3.5196 (1)	X	_	X	X	X	X	4.8864×10^{-2} (1)	X	_	X	X	X	X	1
3 D-GP	5.5616 (4)	X	X	_	_	_	_	4.9281 (3)	X	X	_	_	_	_	2.0899×10^{-1} (3)	X	X	_	X	_	X	3.33
4 C-GP	5.8036 (5)	X	X	_	_	_	_	5.0353 (4)	X	X	_	_	_	_	2.5648×10^{-1} (5)	X	X	X	_	_	_	4.67
5 P-GA	5.6969 (6)	X	X	_	_	_	_	5.1566 (5)	X	X	_	_	_	_	2.5450×10^{-1} (4)	X	X	_	_	_	_	5
6 PNN	5.5449 (3)	X	X	_	_	_	_	5.1725 (6)	X	X	_	_	_	_	2.5935×10^{-1} (6)	X	X	X	_	_	_	4

oP – Average RMAE obtained by the algorithms; rA – rank; ‘X’ – difference between the two algorithms is significant; ‘_’ - difference between the two algorithms is not significant;