International Journal of Innovative
Computing, Information and Control
Volume **10**, Number **4**, August **2014**

 ISSN 1349-4198
pp. 1413–1433

IMPROVEMENT OF CLASSICAL EVOLUTIONARY PROGRAMMING USING STATE FEEDBACK CONTROLLER

Yousef Alipouri[1], Javad Poshtan[1] and Hasan Alipour[2]

[1]Department of Electrical Engineering
Iran University of Science and Technology
Narmak, Tehran 16846-13114, Iran
{ yalipouri; jposhtan }@iust.ac.ir

[2]Department of Electrical Engineering
University of Tabriz
Tabriz 51666-14766, Iran
hasan.alipour2006@gmail.com

Received October 2012; revised October 2013

Abstract. *In evolutionary programming, each parent has two pieces of information: location and cost. The cost of parent specifies whether its location is suitable for breeding offspring or not. If the parent's cost is an acceptable value, producing offspring (or even steering other offspring) in the parent's area is advisable. This information is used in estimating the region of global minimum; then, using the state feedback controller, the offspring is steered to the optimal region. In the proposed method, the cost and coordination of parents have been used for breeding more elite individuals. Many (sixty-five) well-known cost functions have been selected from different references to reveal the pros and cons of our algorithms. In the first stage, the proposed algorithm has been compared inside the EP family. This stage shows promising results for the proposed algorithm. In the second stage, comparison has been performed out of the EP frontiers in which algorithms are state-of-the-art in the optimization field, and are well known inside and outside of their own families. The statistic test has been performed among the algorithms. CPU time and its sensitivity to variable bounds, population and cost function dimensions have been studied. Finally, the proposed method is used in designing the nonlinear minimum variance controller for CSTR (Continuous Stirred-Tank Reactor) benchmark system.*
Keywords: Evolutionary programming, Weighted mean point, State feedback controller, Global optimization, CSTR benchmark system

1. **Introduction.** In the case of evolutionary computation, there are four historical paradigms which have served as the basis for many of the activities in the field: Genetic Algorithms (GA) (Holland and Harbor, 1975) [1], Genetic Programming (GP) (Koza, 1992) [2], Evolutionary Strategies (ES) (Recheuberg, 1973) [3] and Evolutionary Programming (EP) (Fogel et al., 1966) [4]. The basic differences between the paradigms lie in the nature of the representation schemes, reproduction operators and selection methods [5].

In conventional EAs, there is no feedback from the produced or cached information in deciding the coordinate of the new offspring. In addition, in conventional EAs, especially in EP, the region of each parent is the location for breeding the related offspring [6]; thus, the place and cost of other parents are not effective in deciding the coordinate of the offspring. However, the cost of parent specifies whether its location is suitable for breeding offspring or not. Using this information, an algorithm can enhance its performance.

In this paper, an improved version of CEP (Classic EP) is introduced in order to boost EP's performance in searching for the global minimum of multimodal cost functions. The

proposed method attempts to benefit from more information found in each iteration. If the parent's cost is acceptable among others, producing offspring near to its area and even steering other offspring toward that area is advisable. This approach enhances elitism in searching the cost plate.

The proposed algorithm is compared in and out of the evolutionary programming family. Inside the EP family, some well-known algorithms are selected for comparison. Outside the EP family, three groups of families are selected for comparison. Each family has one candidate for comparison. An attempt is made to select the algorithms that are highly capable and similar to the proposed method in methodology and style of searching the global minimum.

The organization of this paper is as follows. Section 2 presents a brief explanation on the background of algorithms. Section 3 introduces the basic ideas behind the Weighted Mean Classical Evolutionary Programming (WMCEP). Section 4 gives some details on test functions, and shows and compares simulation results. Section 5 includes statistic test of "Contrast Estimation of Median" among the algorithms. Section 6 tests sensitivity of CMA-ES (Covariance Matrix Adaptation Evolution Strategy) [7] and WMCEP methods on changes in parameters and compares the methods concerning CPU time. Section 7 presents an application of designing nonlinear minimum variance controller using WMCEP, and finally, Section 8 presents the conclusions and summarizes the simulation results.

2. **Problem Statement and Preliminaries.** The CEP can be implemented as follows [8] (This is the algorithm of EP as implemented in this paper.):

1) Generate the initial population of μ individuals and set $k = 1$. Each individual is taken as a pair of real valued vectors, (x_i, η_i), $\forall i \in \{1, \ldots, \mu\}$, where x_is are objective variables and η_is are standard deviations for Gaussian mutations (also known as strategy parameters in self-adaptive evolutionary algorithms).
2) Evaluate the fitness score for each individual (x_i, η_i), $\forall i \in \{1, \ldots, \mu\}$ of the population based on the objective function $f(x_i)$.
3) Each parent (x_i, η_i), $i = 1, \ldots, \mu$ creates a single offspring (x'_i, η'_i) by: $j = 1, \ldots, n$
4)

$$x'_i(j) = x_i(j) + \eta_i(j) N_j(0, 1) \tag{1}$$

$$\eta'_i = \eta_i(j) \exp(\tau' N(0, 1) + \tau N_j(0, 1)) \tag{2}$$

where $x_i(j)$, $x'_i(j)$, $\eta_i(j)$ and $\eta'_i(j)$ denote the jth component of the vectors x_i, x'_i, η_i and η'_i, respectively. $N(0, 1)$ denotes a normally distributed one-dimensional random number with the mean of zero and standard deviation of one. $N_j(0, 1)$ indicates that the random number is generated anew for each value of j. The factors τ and τ' are commonly set to $(\sqrt{2\sqrt{n}})^{-1}$ and $(2\sqrt{n})^{-1}$.

5) Calculate the fitness of each offspring (x'_i, η'_i), $\forall i \in \{1, \ldots, \mu\}$.
6) Conduct a pairwise comparison over the union of parents (x_i, η_i) and offspring (x'_i, η'_i), $\forall i \in \{1, \ldots, \mu\}$. For each individual, q opponents are chosen uniformly at random from all the parents and offsprings. For each comparison, if the individual's fitness is not smaller than the opponent's, it will receive a "win".
7) Select μ individuals out of (x_i, η_i) and (x'_i, η'_i), $\forall i \in \{1, \ldots, \mu\}$, that have the most wins to be the parents of the next generation.
8) Stop if the halting criterion is satisfied; otherwise, $k = k + 1$ and go to Step 3.

Many variants of EP have attempted to boost performance of the CEP [8-13]. They have usually changed Equations (1) and (2) in Step 3. Yao et al. [8] proposed a Cauchy-mutation-based EP, called fast EP (FEP). Fast EP (FEP) is similar to CEP, but uses a Cauchy function instead of a Gaussian function mutation as the primary search operator in Equations (1) and (2).

LEP (Levy distributed Evolutionary Programming) is a variant of EP, which is similar to CEP and FEP. The difference is in the mutation function; LEP uses a Levy distribution function in place of Gaussian for mutation function (Equation (2)) [9].

All three Gaussian, Cauchy, and Levy distribution functions are special cases of stable distributions. These distribution functions can be produced by Equation (3) in which x and y are Gaussian random numbers.

$$p = \frac{x}{y^{\frac{1}{\alpha}}} \tag{3}$$

$$\text{if } \alpha = 2 \Rightarrow p \text{ is Guassian random number}$$

$$\text{if } \alpha = 1 \Rightarrow p \text{ is Cauchy random number}$$

$$\text{if } 1 < \alpha < 2 \Rightarrow p \text{ is Levy random number}$$

Three algorithms CEP, FEP and LEP are the most famous algorithms inside the EP family and new introduced variants are usually compared with these algorithms. For an in-depth review on EP variants, refer to [10].

3. **The Proposed Method: Weighted Mean Classical Evolutionary Programming (WMCEP).** Conceptually, each individual of one generation can be considered as a person looking for the coordinate of an unknown place (the global minimum). Each individual would want to approach a point, which does not have any previous records, information, or maps. The winner, which is named "the best individual", is the individual who can achieve the best estimation of the coordinate. Individuals are ranked according to their closeness to the global minimum, sorting individuals from low to high cost. At the end of each iteration, after a big tournament between new individuals and parents, some of the individuals are discarded because of their low ranks. It is necessary to say that parents are those who have won the last iteration. This is the approach of CEP, which may be regarded as blind searching.

When the new individual enters the tournament, the logical decision is to use the remaining information from the previous searching team (parents). If parents have found the proper region with low cost, it is advisable to search the same area and recede from areas that have high costs. However, care must be taken for avoiding trapping inside the local minimum. This approach is similar to the method called MCEP (Momentum Coefficient Evolutionary Programming) recently proposed in [6]. However, MCEP uses only the location of parents, and not their costs. One disadvantage of disregarding the cost of parents is pulling both best and worst individuals toward the same region. This means that best individuals do not have the opportunity to search their own area. Our scope in this paper is to add the information of individual costs in determining the mean point by steering offspring toward the best individuals' area (those with best costs) and prevent them from getting close to the worst ones. Figure 1 shows this strategy.

For implementing this idea (Figure 1), two changes have been performed on CEP: 1) Weighted Mean Point (WMP) is defined as an estimation of the global minimum and 2) An approach similar to state feedback is used for steering offspring toward WMP.

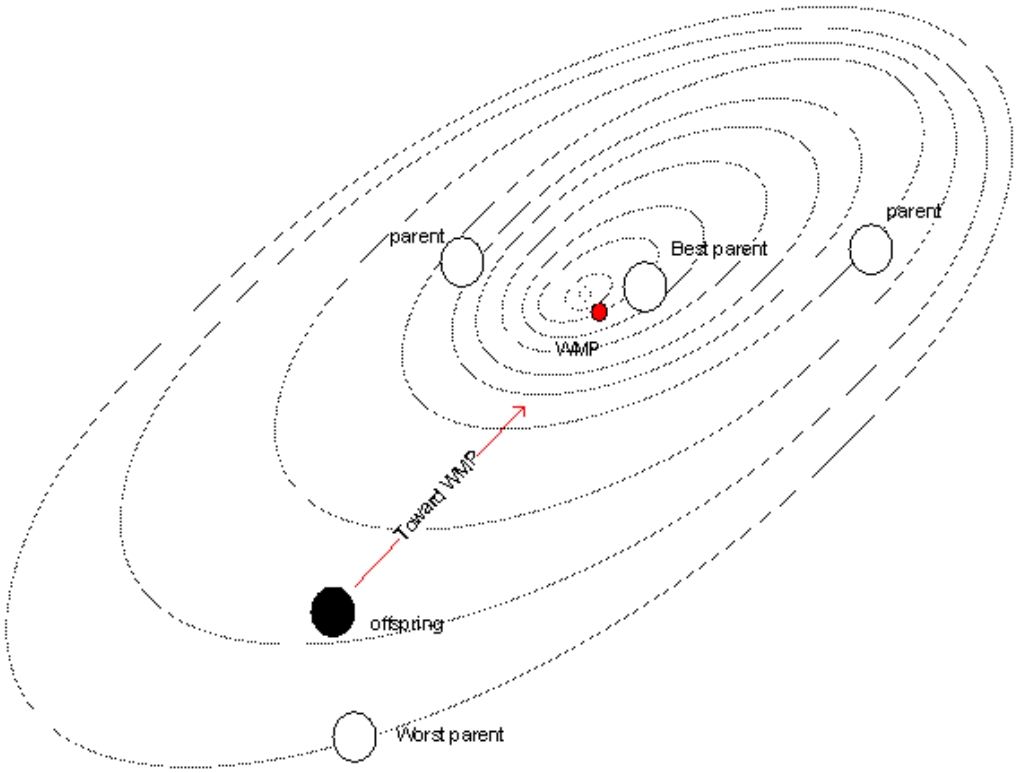

FIGURE 1. Steering the offspring toward Weighted Mean Point (WMP). Ellipses are contours of a cost function with one global minimum.

3.1. **Determining the WMP.** In the proposed method, individuals are ranked using their cost; then a weight is allotted for each individual according to its rank. The individual that has the best cost attains a higher weight than others do, and vice versa. Thus, the proposed method introduces the weighted average of individuals for determining the location of WMP. Equation (4) shows the procedure of calculating WMP. Figure 2 shows a curve of weight versus rank, which is used in this study. Weighting leads WMP closer to the individuals with higher ranks. Therefore, they have more shares in determining the place of WMP. Steering the offspring toward WMP causes the offspring to move toward the best individuals, and prevents them from getting close to the parents with low ranks (Figure 1). By increasing the slope of curve in Figure 2, WMCEP becomes more elite, which may cause it to be trapped inside the local minimums.

$$Wmean = \sum_{i=1}^{n} w_i x_i, \quad \sum_{i=1}^{n} w_i = 1, \quad w_1 \geq w_2 \geq \ldots \geq w_\mu > 0 \tag{4}$$

where

$$w_i = \frac{e^{a(i)}}{\sum_{i=1}^{n} e^{a(i)}}$$

$$a\,(i) = 3 - \frac{i \times 6}{n}$$

where n is the population size.

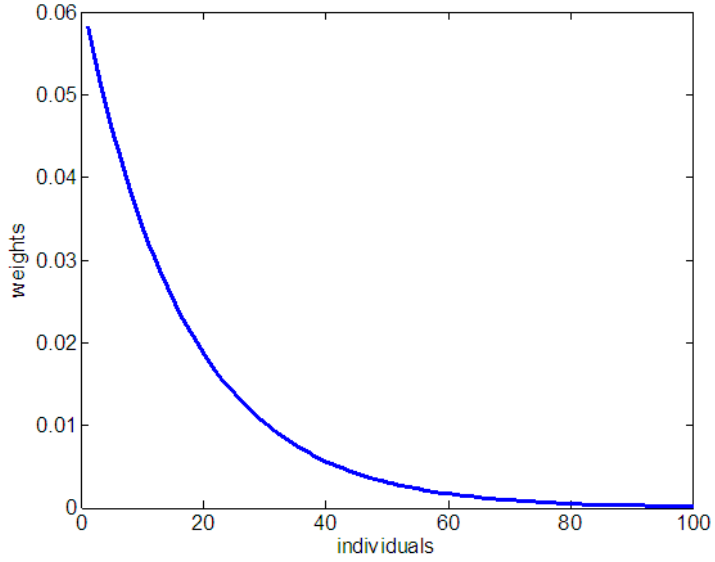

FIGURE 2. Curve of weights via the individual's rank

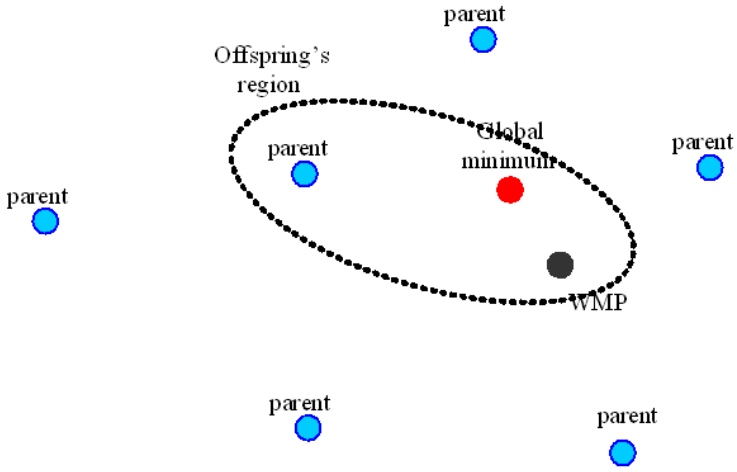

FIGURE 3. WMCEP extends the offspring region by steering the offspring toward WMP

WMCEP extends the offspring region by steering the offspring toward WMP. Figure 3 shows the breeding region in the WMCEP algorithm. This region is larger than the region defined for CEP (see Figure 4). Thus, it is more probable that the global minimum is located inside the WMP region (depicted in Figure 3) and the probability of reaching the global minimum is more than other EP variants (see Figure 4). Figure 5 shows the offspring that was produced in the region near WMP.

Figure 5 shows that steering toward WMP can increase the convergence speed and accuracy of the method. Nevertheless, producing all offspring near WMP can lead to falling inside local minimums. Thus, the breading region must be controlled. The region near the parent is more robust against falling into local minimums, because in this region the step size for changing the offspring is small and most offspring are produced near their own parents; however, searching speed is very low (see Figure 4). On the other hand, the region near WMP gives a big step size for the offspring and, as a result, increases the

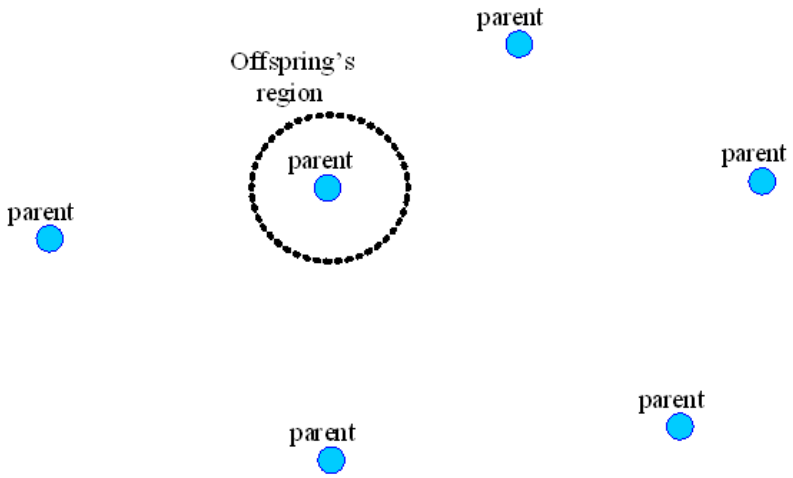

FIGURE 4. In the conventional EP, offspring is produced using the information of its own parent [6]

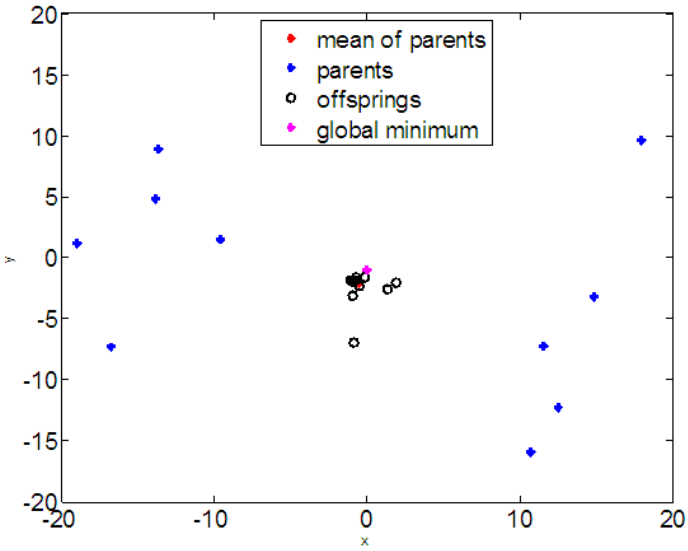

FIGURE 5. The results of WMCEP in producing offspring by steering them toward the weighted mean point of the parents

speed of the algorithm; however, it may cause trapping inside the local minimum, because most of the offsprings are produced in a small area near to WMP (see Figures 2 and 5). Consequently, the optimal region must be selected between parents and WMP regions. State feedback determines this optimal region.

3.2. **Designing state feedback controller.** WMCEP faces two challenges: 1) The weighted mean point may be far from the global minimum in the last iterations (if it falls in the local minimum, then it will mislead the offspring), and 2) speeding up the algorithms can cause premature convergence. The method that is more capable has more speed while it avoids being trapped in local minimums.

Using information has been obtained from the previous iteration can simultaneously fulfill both criteria of speed and accuracy. For solving both introduced issues, the speed

of the algorithm must be controlled while getting close to the WMP. A controller similar to state feedback controls regions for breeding.

State feedback is a terminology used in control engineering (see block diagram in Figure 6).

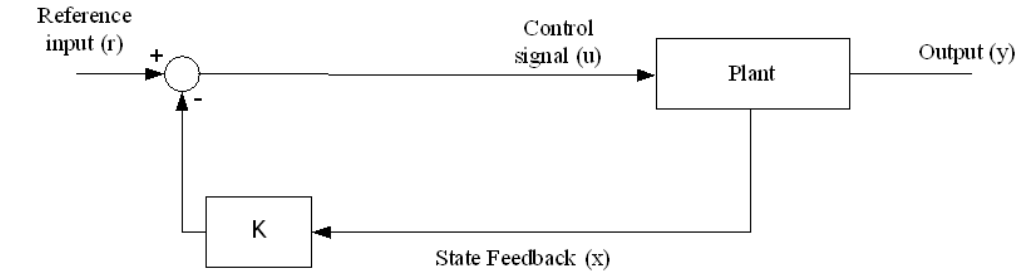

FIGURE 6. Block diagram of the state feedback controller

State feedback is a kind of controller in which states of the system are measured and the control signal is produced by feedback gain. In state feedback, the value of the state vector is fed back to the input of the system. State feedback has an input, r, and defines the following relationship [11]:

$$u(t) = r(t) + Kx(t) \tag{5}$$

K is a constant matrix that is external to the system.

The searching process of EAs can be defined using the control block diagram. Figure 7 shows the searching scheme of conventional EAs which is similar to the open loop configuration. In this scheme, there is no control and feedback on the searching process.

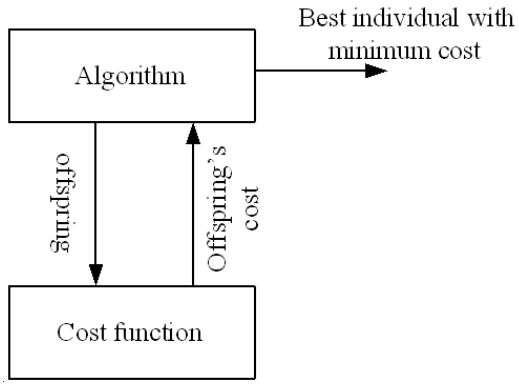

FIGURE 7. Block diagram of conventional EAs searching process

WMP is far from the global minimum in the first iterations, but gets closer to it over time provided the algorithm be not trapped in local minimum. Hence, WMP can be used as the estimated coordinate of the global minimum. This estimation can be used as a basis for drawing feedback and designing the state feedback controller. Figure 8 shows the diagram of this process.

Similar to Equation (5), the controller similar to state feedback can be defined for WMCEP. Therefore, the offspring can be produced by Equation (6).

$$x'_i(j) = x_i(j) + \eta_i(j)N_j(0,1) + K \times WMP(j) \tag{6}$$

where the $WMP(j)$ is the j element of WMP coordination. Now, the value of k must be determined. In state feedback, the value of K is extremely dependent on the system.

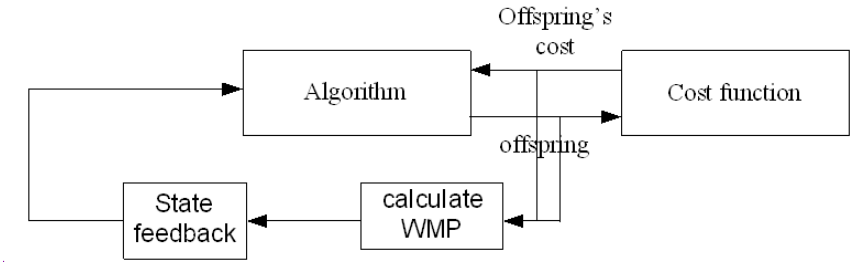

FIGURE 8. Block diagram of the state feedback control used in WMCEP

Consequently, the value of K in Equation (6) depends on the shape of cost functions. This dependence damages the generality of the algorithm. However, this value can be designed in such a way to be suitable for most kinds of cost functions (Equations (7) and (8)). Coefficient S is used for avoiding producing points out of the variable's boundaries. Thus, consider $S + K = 1$. Then,

$$x_i'(j) = S \times x_i(j) + \eta_i(j) N_j(0,1) + K \times Wmean(j) \tag{7}$$

$$S = \frac{(total\ iter - iter)}{total\ iter}, \quad K = \frac{iter}{total\ iter} \tag{8}$$

In Equation (8), *total iter* is the abbreviation for the total number of iterations, which is specified by the user, and *iter* is the abbreviation for the current iteration.

Equation (8) introduces descending and ascending coefficients S and K. Defining S and K by (8) has some advantages. In the first iterations, the algorithm does not have any estimation of the global minimum, so as far as WMP is considered as the initial destination, the offspring can be given a known target. Consequently, speed can be increased by steering offsprings toward this destination. This is a goal of introducing WMP, but over time, the algorithm can find its way toward the global minimum. Therefore, by defining the descending coefficient for the weighted mean value (K in Equation (8)); the effectiveness of the WMP will be decreased. This strategy enables the algorithm to change its way toward the global minimum after finding good estimation of the global minimum. In other words, over time, the necessity for the weighted mean value decreases and the algorithm is able to find points near the global minimum.

At first glance, it may seem that this strategy is only possible when the global minimum is in the center of the search plate. However, searching with a high number of individuals helps in contemplating on all parts of the map simultaneously. The results of this paper indicate that this method is capable of finding the global minimum located anywhere on the search map.

In brief, WMCEP has two main differences with CEP: 1) Giving weights to the individuals by their rank, and defining WMP, which is the weighted average of the individual's location, and 2) Controlling the region of breeding offspring with regard to WMP. Therefore, the pseudo code of WMCEP is as follows.

Choose the initial population of individuals
Produce strategy parameters by any mutation function
Evaluate the cost of each individual in that population
Repeat this generation until termination: (time limit, sufficient cost achieved, etc.)
    Rank individuals using their cost and give them weights
    Update the weighted mean point using (4)
    Breed individuals through mutation (Equation (7)) to give birth to the offspring
    Evaluate the cost of the offspring

Raise tournament to decide the next generation members

End

Figure 9 shows the flowchart of WMCEP.

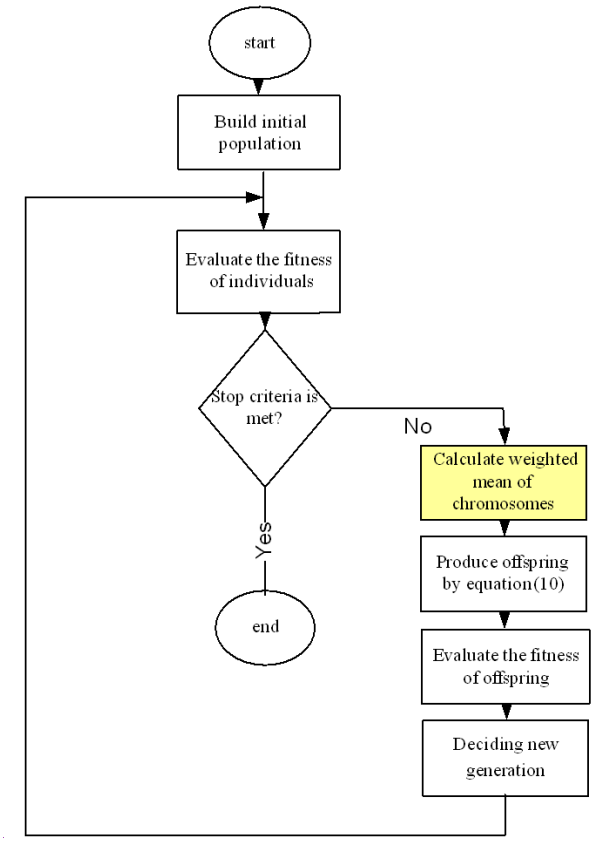

FIGURE 9. Flowchart of WMCEP

4. **Results.** Nine algorithms of CEP, FEP [8], LEP [9], EEP (Exponential Evolutionary Programming) [12], RLEP (Evolutionary Programming based on Reinforcement Learning) [13], CMA-ES [7], JG-GA (Jumping Gene GA) [14,15], IW-PSO (Increasing Inertia Weight PSO) [16] and WMCEP are compared in this section. These algorithms consist of two groups: algorithms are and are not considered as evolutionary programming variants. An attempt has been made to select the algorithms, which are known in and out of their own families and have some similarities to the proposed method. No attempt was made to optimize algorithms against cost functions.

Table 1 shows the name and references of cost functions. Their parameters, dimensions and variable bounds are the same as the references. The source codes of these cost functions are available in [17]. A detailed explanation of cost functions can be found in the references cited in Table 1. All parameters of the algorithms are the same as introduced in Table 2.

The algorithms have been tested on the 65 cost functions. Each test has been repeated 50 times and the average results are shown in Table 3. This table compares the accuracy of the methods in obtaining the global minimum.

Distances between the optimal points found by WMCEP and other algorithms show that the performance of the proposed method is notable. This can be an effect of implementing the strategy of using information of the best individual's costs and locations in producing offspring.

TABLE 1. Description of the sixty-five cost functions used in this study

Cost function	Name	Dimension (n)	Bounds	Global minimum	Reference
F1	*Sphere Model*	10	$[-10, 10]^n$	-92.65	[18]
F2	*Ellipsoidal Function*	10	$[-10, 10]^n$	276.32	[18]
F3	*Rastrigin Function*	10	$[-10, 10]^n$	20.91	[18]
F4	*Buche-Rastrigin Function*	10	$[-10, 10]^n$	20.91	[18]
F5	*Linear Slope*	10	$[-10, 10]^n$	51.53	[18]
F6	*Attractive Sector Function*	10	$[-10, 10]^n$	83.48	[18]
F7	*Step Ellipsoidal Function*	10	$[-10, 10]^n$	-83.87	[18]
F8	*Rosenbrock Function, original*	10	$[-10, 10]^n$	-135	[18]
F9	*Rosenbrock Function, rotated*	10	$[-10, 10]^n$	-359	[18]
F10	*Ellipsoidal Function*	10	$[-10, 10]^n$	-78.98	[18]
F11	*Discus Function*	10	$[-10, 10]^n$	-101	[18]
F12	*Bent Cigar Function*	10	$[-10, 10]^n$	295.1	[18]
F13	*Sharp Ridge Function*	10	$[-10, 10]^n$	-51.73	[18]
F14	*Different Powers Function*	10	$[-10, 10]^n$	-57.90	[18]
F15	*Rastrigin Function*	10	$[-10, 10]^n$	-44.77	[18]
F16	*Weierstrass Function*	10	$[-10, 10]^n$	-260	[18]
F17	*Schaffers F7 Function*	10	$[-10, 10]^n$	-38.72	[18]
F18	*Schaffers F7 Function, moderately ill-conditioned*	10	$[-10, 10]^n$	-38.72	[18]
F19	*Composite Griewank-Rosenbrock Function F8F2*	10	$[-10, 10]^n$	40.46	[18]
F20	*Schwefel Function*	10	$[-10, 10]^n$	183.1	[18]
F21	*Gallagher's Gaussian 101-me Peaks Function*	10	$[-10, 10]^n$	310.6	[18]
F22	*Gallagher's Gaussian 21-hi Peaks Function*	10	$[-10, 10]^n$	42.97	[18]
F23	*Katsuura Function*	10	$[-10, 10]^n$	210.4	[18]
F24	*Lunacek bi-Rastrigin Function*	10	$[-10, 10]^n$	47.56	[18]
F25	*Generalized Schwefel's Problem 2.26*	30	$[-500, 500]^n$	-12569.5	[8]
F26	*Generalized Rastrigin's Function*	30	$[-5.12, 5.12]^n$	0	[8]
F27	*Ackley's Function*	30	$[-32, 32]^n$	0	[8]
F28	*Generalized Griewank Function*	30	$[-600, 600]^n$	0	[8]
F29	*Generalized Penalized Functions*	30	$[-50, 50]^n$	0	[8]
F30	*Generalized Penalized Functions2*	30	$[-50, 50]^n$	0	[8]
F31	*Shekel's Foxholes Function*	2	$[-65.53, 65.53]^n$	1	[8]
F32	*Kowalik's Function*	4	$[-5, 5]^n$	0.0003075	[8]
F33	*Six-Hump Camel-Back Function*	2	$[-5, 5]^n$	-1.0316285	[8]
F34	*Branin Function*	2	$[-5, 10] \times [0, 15]$	0.398	[8]
F35	*Goldstein-Price Function*	2	$[-2, 2]^n$	3	[8]
F36	*Hartman's Family1*	3	$[0, 1]^n$	-3.86	[8]
F37	*Hartman's Family2*	6	$[0, 1]^n$	-3.32	[8]
F38	*Shekel's Family1*	4	$[0, 10]^n$	-10	[8]
F39	*Shekel's Family2*	4	$[0, 10]^n$	-10	[8]
F40	*Shekel's Family3*	4	$[0, 10]^n$	-10	[8]
F41	*Corana's parabola*	4	$[-1000, 1000]^n$	0	[19]
F42	*deceptive function*	10	$[0, 1]^n$	-16	[19]
F43	*Sum of different power*	30	$[-1, 1]^n$	0	[20]
F44	*Beale function*	10	[-5,10]^n	0	[20]
F45	*Alpine function*	10	$[-10, 10]^n$	0	[20]
F46	*Inverted cosine wave function (Masters)*	10	$[-5, 5]^n$	$-n + 1$	[20]
F47	*Hyper-Ellipsoid*	10	$[-100, 100]^n$	0	[21]
F48	*Neumaier #3*	30	$[-900, 900]^n$	-4930	[21]
F49	*Salomon*	10	$[-10, 10]^n$	0	[21]
F50	*Lennard-Jones*	15	$[-2, 2]^n$	$-$	[21]
F51	*Odd Square*	20	$[-5\pi, 5\pi]^n$	-1.14383	[21]
F52	*Katsuura*	10	$[-1000, 1000]^n$	1	[21]
F53	*Bohachevsky 1 Problem (BF1)*	2	$[-50, 50]^n$	0	[22]
F54	*Camel Back – 3 Three Hump Problem (CB3)*	2	$[-5, 5]^n$	0	[22]
F55	*Cosine Mixture Problem (CM)*	10	$[-100, 100]^n$	$-$	[22]
F56	*Easom Problem (EP)*	2	$[-10, 10]^n$	-1	[22]
F57	*Epistatic Michalewicz Problem (EM)*	10	$[0, \pi]^n$	-9.660152	[22]
F58	*Exponential Problem (EXP)*	30	$[-1, 1]^n$	-1	[22]
F59	*Meyer and Roth Problem (MR)*	3	$[-10, 10]^n$	0.00004	[22]
F60	*Modified Rosenbrock Problem (MRP)*	2	$[-5, 5]^n$	0	[22]
F61	*Multi-Gaussian Problem (MGP)*	2	$[-2, 2]^n$	1.29695	[22]
F62	*Paviani Problem (PP)*	10	$[2, 10]^n$	-45.778	[22]
F63	*Schaffer 2 Problem (SF2)*	2	$[-10, 10]^n$	0	[22]
F64	*Shubert Problem (SBT)*	2	$[-10, 10]^n$	-186.7309	[22]
F65	*Sinusoidal Problem (SIN)*	10	$[0, 180]^n$	-3.5	[22]

Table 2. Parameters of evolutionary programming's variants

General	Population size	100
	Number of repetition	50
	Tournament size q	10
	Initial standard deviation	3
FEP	Parameter t for Cauchy distribution function	1
LEP	Value of α for Levy distribution function in Equation (3)	1.5
CMA-ES	All parameters are similar to the source codes in [8]	
JG	Number of transposon	1
	Length of transposon	2
	crossover	Uniform
	Mutation rate	0.1
IW-PSO	Acceleration coefficients	2
	Linearly increasing inertia weight	From 0.5 to 1.5
	Maximum velocity	$\pm X_{max}$

It must be noticed that the RLEP is a fast and accurate algorithm, which produces offspring by predicting the future performance of the algorithm. However, it needs many calculations and evaluates many cost functions (at least four times more than CEP). Therefore, in all comparisons of this paper, all the algorithms were run until the prespecified number of cost functions was evaluated.

5. **Statistical Test.** In recent years, use of statistical tests for improving performance evaluation of a new method has become a widespread technique in computational intelligence. In this section, a procedure is assigned to estimate the differences between several algorithms. It is named the Contrast Estimation of Medians method. This method is very recommendable if assumed that the global performance is reflected by the magnitudes of differences between performances of the algorithms [23]. These estimators can be understood as an advanced global performance measure. It is especially useful to estimate the extent to which an algorithm outperforms another one [23].

In the current experimental analysis, the set of estimators of medians is calculated directly from the average error results. Table 8 shows the estimations computed for each algorithm. This comparison was performed in two groups: inside and outside of the EP family. By focusing attention on the rows of the table, the performance of WMCEP may be highlighted (all its related estimators are negative, i.e., it achieves very low error rates considering median estimators). On the other hand, FEP achieves higher error rates in this experimental study. Table 4 shows that WMCEP is most similar to CMA-ES in error rate (performance), as the Contrast estimated method shows small difference between error rate of WMCEP and CMA-ES. Thus, in next step, WMCEP and CMA-ES are compared in terms of performance in more detail.

6. **Comparing WMCEP and CMA-ES.** In this section, the two algorithms of WMCEP and CMA-ES are compared in detail. CPU times and their sensitivities to variable bounds, number of population and cost function dimensions are the issues of this section.

6.1. **CPU time.** It is predictable that CMA-ES requires more CPU time. This algorithm has more complicated equations and needs many calculations [7]. Two algorithms, WMCEP and CMA-ES, were tested on a 30 dimensional cost function. An iteration of CMA-ES needs, in average, 0.0272 seconds, while WMCEP requires 0.012 seconds.

TABLE 3. Results of 50 repetitions of testing algorithms on 65 cost functions

	CEP	FEP	LEP	EEP	RLEP	CMAES	IIW-PSO	GA-JG	WMCEP	Global	Generation
F1	525.0	−18.63	−37.83	−92.17	−91.70	**−92.64**	−90.94	−88.24	**−92.64**	−92.65	100
F2	27634	25192	70705	1016.9	515.24	**276.32**	11728	3728	**276.32**	276.32	100
F3	723	70.60	57.60	42.75	37.73	25.08	111.8	94.32	**23.29**	20.91	500
F4	53.96	906.2	35.19	**24.88**	27.11	30.85	133.6	117.2	29.77	20.91	500
F5	**51.53**	**51.53**	**51.53**	**51.53**	51.97	62.52	53.92	53.41	**51.53**	51.53	50
F6	2189	1883	86.06	102.4	85.46	**83.48**	144.7	112.3	**83.49**	83.48	500
F7	5052	1516	1.501	−61.07	−64.27	**−83.87**	−54.94	−63.69	−82.22	−83.87	100
F8	−10.29	1402	−3.37	−127.6	−120.8	**−135**	−113.2	153.5	−128.5	−135	500
F9	−349	800231	3646	−268.3	−250	**−359.2**	−318.1	−324.5	−351.2	−359	200
F10	80977	48179	70459	3940.2	886.51	**−78.98**	538.5	7664.5	975	−78.98	5000
F11	−7.05	20.29	−33.01	52.92	14.68	**−101**	−92.82	−56.44	−59.77	−101	1000
F12	342	11742	337	379.8	7225.8	**295.1**	341.8	1.8e + 6	299.6	295.1	500
F13	−6.32	17.77	−34.99	−48.47	−28.21	**−51.73**	−24.41	157	−51.15	−51.73	1000
F14	−57.87	−57.07	−57.84	−57.83	−56.40	**−57.9**	−57.87	−56.40	**−57.9**	−57.90	200
F15	−20.57	9.955	−30.84	−15.91	−36.59	**−42.87**	7.067	27.61	−41.57	−44.77	1000
F16	−257.5	−252.4	−257.4	−258.3	−258.4	−256.2	−254.9	−252.8	**−259.3**	−260	1000
F17	−37.25	−36.11	−38.29	−38.15	−38.22	**−38.72**	−36.59	−36.65	−38.60	−38.72	200
F18	−37.11	−29.53	−37.97	−37.89	−36.02	**−38.72**	−33.88	−33.46	−37.60	−38.72	500
F19	42.40	42.19	42.09	41.54	42.63	**40.63**	42.80	42.54	42.21	40.46	300
F20	1.842	215.8	**184**	**184**	185.2	**184**	**184**	186	**184**	183.1	200
F21	**310.6**	311.5	**310.6**	**310.6**	311.8	317	318	322	**310.6**	310.6	500
F22	53.33	46.54	**45.18**	52.57	45.78	58.10	57.19	52.22	47.96	42.97	500
F23	211.7	**210.5**	211.7	211.7	211.7	**210.5**	210.8	211.5	211.4	210.4	3000
F24	67.57	107.4	72.81	78.83	71.22	56.837	77.63	100	**56.318**	47.56	1000
F25	−8455	**−12493**	−12351	−11026	−11878.2	−2572	−8917	−7040	−8384	−12569.5	3000
F26	57.01	6.76	98.59	369.6	106.8	8.06	73.77	103	**5.57**	0	3000
F27	6.36	1.13	0.26	20.01	1.842	**8e − 15**	5	5.14	5e − 12	0	3000
F28	8.16	1.63	0.44	3.31	0.928	**0**	1.03	2.32	**0**	0	3000
F29	15.65	0.66	0.002	2.42	0.149	21.74	18.52	3.32	**1.54e − 24**	0	3000
F30	62.60	10.32	0.16	86.38	0.020	1.20	31.6	13.30	**9.7e − 26**	0	3000
F31	**1**	**1**	**1**	1.246	**1**	2.98	**1**	**1**	**1**	1	2000
F32	0.006	0.01	0.005	0.003	0.0056	0.01	0.02	0.006	**0.004**	0.0003075	400
F33	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	−0.99	**−1.031**	−1.030	**−1.031**	−1.0316285	100
F34	**0.397**	**0.397**	**0.397**	**0.397**	**0.397**	**0.397**	**0.397**	**0.397**	**0.397**	0.397	100
F35	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	3	100
F36	**−3.86**	**−3.86**	**−3.86**	**−3.86**	**−3.86**	**−3.862**	**−3.862**	**−3.862**	**−3.862**	−3.86	100
F37	−3.298	−3.298	−3.274	−3.25	**−3.32**	−3.31	−3.28	−3.28	−3.26	−3.32	500
F38	**−10.15**	**−10.15**	**−10.15**	−7.17	**−10.15**	−5.74	−8.13	−7.31	−8.2	−10	100
F39	−9.50	−8.82	−9.01	−7.86	−8.91	−6.58	−8.43	−9.26	**−9.47**	−10	100
F40	−9.15	−8.71	−9.90	−8.55	−9.60	−5.73	−7.41	−9.59	**−10.52**	−10	100
F41	**0**	**0**	**0**	**0**	**0**	3.957	**0**	**0**	**0**	0	1000
F42	**−16**	−15.81	−12.33	**−16**	−13.55	−5.12	**−16**	−13.65	**−16**	−16	1000
F43	2.76e − 39	1.09e − 18	0.003	3.4e − 34	5.21e − 5	5.1e − 60	1.1e − 12	1.54e − 8	**1.1e − 82**	0	3000
F44	7.15e − 6	1.15	0.003	0.0011	0.0010	**1e − 126**	3.8e − 4	0.06	1e − 07	0	1000
F45	4.38e − 15	0.016	5e − 16	4.1e − 15	2.54e − 16	3.2e − 14	0.1	0.007	**6.9e − 19**	0	1000
F46	−7.191	−8.333	−8.257	**−8.341**	**−8.341**	−5.16	−6.90	−8.15	**−8.34**	$-n+1$	1000
F47	6.53e − 12	14.67	3e − 13	9.61e − 6	3.79e − 6	**1e − 121**	33.71	26.84	1.1e − 43	0	1000
F48	270736	51291	18935	127880	5951	**−4930**	15485	21654	−1053	−4930	3000
F49	0.195	0.980	0.190	0.599	0.199	0.14	0.48	0.22	**0.10**	0	1000
F50	**−5.288**	−5.286	**−5.288**	−5.11	−4.07	**−5.28**	**−5.28**	−4.73	**−5.28**	–	1000
F51	−0.03	−3.2e − 13	−0.06	−2.3e − 7	−0.085	**−0.74**	−0.001	−2.5e − 4	−0.63	−1.14383	2000
F52	**1**	7.07	**1**	4.093	**1**	**1**	5e + 11	3.4e + 10	**1**	1	1000
F53	5.4e − 13	0.032	5e − 12	5.5e − 17	2.96e − 7	**0**	2e − 5	0.08	**0**	0	100
F54	2.03e − 15	2.7e − 06	2e − 13	8.2e − 16	8.72e − 9	0.1	3e − 8	5.4e − 5	**1.1e − 26**	0	100
F55	**−10e + 5**	−99620	−99987	−92418	**−10e + 5**	−65299	**−99999**	−84095	**−99999**	–	1000
F56	**−1**	**−1**	**−1**	**−1**	**−1**	**−1**	**−1**	**−1**	**−1**	−1	100
F57	−7.287	−7.649	−5.921	−7.38	−7.724	−4.4	−5.77	−5.47	**−7.67**	−9.660152	1000
F58	−0.98	−0.974	−0.17	−0.97	−0.55	**−1**	−0.98	−0.97	**−1**	−1	1000
F59	0.0195	0.0021	0.0020	**0.00190**	**0.00190**	0.006	**0.002**	**0.002**	**0.002**	0.00004	300
F60	**1.23e − 32**	2.56e − 30	**1.23e − 32**	**1.23e − 32**	**1.23e − 32**	**1.23e − 32**	4e − 24	1.76e − 6	**1.23e − 32**	0	100
F61	−1.288	−1.279	−1.282	−1.286	−1.295	−1.22	**−1.296**	−1.261	**−1.296**	1.29695	200
F62	**−45.77**	−45.55	**−45.77**	**−45.77**	**−45.77**	**−45.77**	**−45.77**	−45.31	**−45.77**	−45.778	1000
F63	0.181	0.11	1.4e − 17	1.4e − 22	2.49e − 9	3.30	1.1e − 5	0.2	**2.1e − 34**	0	500
F64	**−186**	**−186**	**−186**	**−186**	**−186**	−185.4	−186.6	−186.5	**−186.7**	−186.730	100
F65	**−3.5**	**−3.5**	**−3.5**	**−3.5**	**−3.5**	**−3.5**	−2.951	−2.42	**−3.5**	−3.5	5000

*Notation e $-a$ means 10^{-a}.

TABLE 4. Contrast estimation method results. Estimators highlight WMCEP as the best performing algorithm.

	CEP	FEP	LEP	EEP	RLEP	WMCEP
CEP	0	−2.23	−1.03	−1	0.57	0.59
FEP	2.23	0	1.19	1.23	2.81	2.82
LEP	1.03	−1.19	0	0.033	1.61	1.63
EEP	1	−1.23	−0.033	0	1.58	1.59
RLEP	−0.57	−2.81	−1.61	−1.58	0	0.016
WMCEP	**−0.59**	**−2.82**	**−1.63**	**−1.59**	**−0.016**	**0**

	CMA-ES	IIW-PSO	GA-JG	WMCEP
IIW-PSO	0	−76	−5.2	547e − 8
GA-IG	76	0	71.6	71
CMA-ES	5.2	−71.6	0	5.2
WMCEP	**−547e −8**	**−71**	**−5.2**	**0**

Therefore, the required time for CMA-ES to evaluate each iteration is 2.26 times more than that of WMCEP. This difference is very crucial in real world applications where cost functions require many iterations in order to reach the global minimum.

6.2. **Variable bounds.** Some applications force bounds on their parameters or coefficients, which must be adapted. As EAs search the map with a population, it is possible that some variables of individuals are produced out of frontiers. In order to fix this problem, two decisions have usually been made. In the first one, the variable set to the frontier value when it comes out. In the second decision, the variable set to the uniform random number inside the searching map. The first decision for fixing this problem makes the CMA-ES lose its destination (Figure 11). The second decision results in oscillation in the algorithm output (Figure 10). It can be seen that the proposed method is not sensitive to boundary limits, because by the normalized coefficient in Equation (8), the algorithm prevents from producing offsprings out of boundaries.

6.3. **Variable numbers.** Table 6 compares performances of CMA-ES and WMCEP in a higher number of variables on five functions while bound limitation is forced on both algorithms. In part IV, bounds were not imposed on CMA-ES when the global minimum was inside the searching map and there was no other better minimum outside the bounds. However, in some cost functions like f25, there are deeper minimums out of bounds, so the algorithm is forced to search inside the searching map. Thus, oscillation in results of some algorithms (like CMA-ES) in Table 3 is due to this decision. It can be seen in Table 5 that the CMA-ES is sensitive to number of variables, but WMCEP is not.

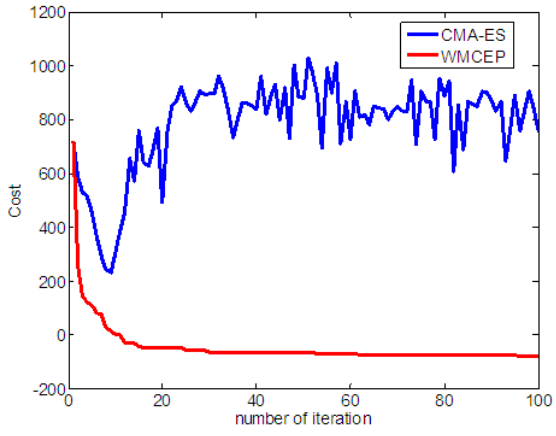

FIGURE 10. Setting the value of variable to the frontier value

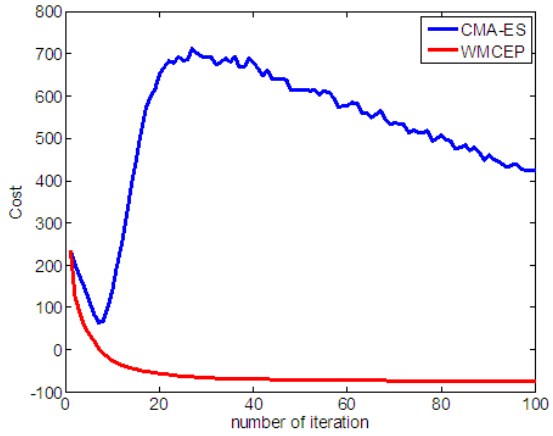

FIGURE 11. Set uniform random value to the variable

TABLE 5. Test sensitivity of CMA-ES and WMCEP against forcing bound limitations for variables

	Dimension	Generation	CMA-ES	WMCEP	Dimension	Generation	CMA-ES	WMCEP	Global minimum
F1	40	400	**863**	**−91.28**	10	100	**−92.64**	**−92.64**	−92.65
F2	40	1000	**276.32**	**276.32**	10	100	**276.32**	**276.32**	276.32
F3	40	1000	**2425**	**153**	10	500	**25.08**	**23.29**	20.91
F4	40	1000	**62.20**	**151**	10	500	**30.85**	**29.77**	20.91
F5	40	100	**564**	**51.53**	10	50	**62.52**	**51.53**	51.53

6.4. **Population size.** Table 6 compares the two algorithms CMA-ES and WMCEP with a certain population. Five functions were selected for this comparison. Tournament size (q) in WMCEP is proportional to the population and is calculated by the equation $q = \frac{10\times\text{population}}{100}$.

TABLE 6. Test sensitivity of population size for CMA-ES and WMCEP

Function	Population	Generation	WMCEP	CMA-ES	Global minimum
F1	20	500	**−92.65**	**−92.65**	−92.65
	50	200	**−92.65**	**−92.65**	−92.65
	100	100	**−92.65**	**−92.65**	−92.65
	200	50	**−92.62**	**−92.64**	−92.65
F2	20	500	**276.32**	**276.32**	276.32
	50	200	**276.32**	**276.32**	276.32
	100	100	**276.32**	**276.32**	276.32
	200	50	**297.62**	**276.55**	276.32
F3	20	1500	**22.89**	**25.88**	20.91
	50	600	**24.193**	**26.282**	20.91
	100	300	**25.08**	**23.29**	20.91
	200	150	**20.97**	**21.90**	20.91
F4	20	2500	**40.11**	**34.59**	20.91
	50	1000	**30.85**	**31.25**	20.91
	100	500	**29.77**	**30.85**	20.91
	200	250	**25.93**	**26.87**	20.91
F5	20	250	**51.53**	**51.53**	51.53
	50	100	**51.53**	**51.53**	51.53
	100	50	**51.53**	**51.53**	51.53
	200	25	**51.53**	**51.53**	51.53

In this part, the number of recalling cost functions for all different populations is consistent. As can be seen, both algorithms were slightly sensitive to population size. It is predictable that increasing the number of individuals in each generation gives them more information about better and worse places for breeding. Therefore, WMP will be more accurate in estimating the global minimum.

7. **Application.** The MVC (Minimum Variance Controller), also referred to as optimal H2 control and first derived in [24] by Astrom (1970) and Box and Jenkins (1970) [25], is the best possible feedback control for linear systems in the sense that it achieves the smallest possible closed-loop output variance [26]. When the model of the process are known, or can be identified, MV controller can be designed to minimize the variance of output. Although almost all real-world systems are nonlinear in behavior, most of the introduced methods for estimating minimum variance use linear approaches and require linear models [27]. MV controllers can be enhanced to deal with nonlinear systems if nonlinear models are considered. However, designing nonlinear MVC especially using Neural Network (NN-MVC) has some drawbacks:

1) In order to design MVC, the explicit relations between outputs and inputs must be executable. This relation is defined implicitly in the nonlinear models [28,29]. Suppose that the plant can be described by the following model,

$$y(t+1) = f(y(t), y(t-1), \ldots, y(t-n_y), u(t), u(t-1), \ldots, y(t-n_u)) \tag{9}$$

where $f(.)$ is a neural network, in which y_t and u_t denote the output and the input signal vectors respectively. Inverse models of systems are necessary for designing MVC. The input-output relation of the neural network modeling the plant inverse is:

$$u(t) = f^{-1}(y(t+1), y(t), y(t-1), \ldots, y(t-n_y), u(t-1), \ldots, y(t-n_u)) \tag{10}$$

Obtaining inverse of neural network has some difficulties such as: **a)** The actual operational inputs may be hard to define a priori; **b)** If a nonlinear system is not one-one, then an incorrect inverse can be obtained; Finding inverse of nonlinear neural network is a tedious task (if possible), and **c)** Control signal is not in hand before obtaining inverse function, and the inverse function is not executable before having control signal (unless function $f(.)$ is known or can be estimated).

These problems cause that researchers define approximated form of neural networks for NN-MVC designing. Some researchers have suggested the affine model of neural networks in which the relation between input and output can be defined explicitly as follows [28]:

$$y(t+1) = f(X(t)) + g(X(t))u(t) + e(t) \tag{11}$$

where $e(t)$ is model mismatch and

$$X(t) = [y(t), y(t-1), \ldots, y(t-n_y), u(t), u(t-1), \ldots, y(t-n_u)]$$

In addition, the linear approximation has been used in literature such as bilinear approximation of Equation (9) that may be parameterized as [29]:

$$y(t+1) = Ay(t) + Bu(t) + g(.) \tag{12}$$

where an assumption is made that B is non-zero. A and B are unknown diagonal parameter matrices and $g(.)$ is the nonlinear part of the equivalent model.

These models are not accurate and are not universal models. These methods can be used in designing many kinds of controllers, but considering that the MVC is an optimal controller that must be accurate in reaching minimum variance, it can be a big disadvantage for these approximated methods. Even, these methods can be less accurate than linear MV methods.

2) Another drawback of using neural network is training issue. Training neural networks is a high dimensional-multimodal optimization task and weight space can be extremely rugged and has many local minima. This problem is avoiding using full neural network model directly as a controller.

The strategy of this paper is to use evolutionary algorithms for weight optimization in domains where gradient methods cannot be directly applied, or where gradient methods are less effective than in simple supervised learning applications.

In this paper, for the first time, EA has been used for designing MVC by training full neural network model. Based on authors' information, the proposed method is a new idea and has not been proposed in previous published literatures. In this regard, WMCEP is used for optimizing the neural network weights. Obviously, a search method that cannot escape from local minima will have difficulty in finding an optimal solution. This method can fill up the gap between advantages of NN-MVC and difficulty in obtaining $f^{\square 1}(.)$.

Consider the neural model for the controller is represented by a two-layer artificial neural network. The neural networks are then trained by WMCEP for a proper number of iterations. After training is completed, the controller is applied on loop, and the output variance is calculated for each individual and all the neural networks are ranked based on the values of their fitness indexes. Figure 12 shows the flowchart of the mentioned WMCEP procedure when the NN-MVC design is adopted.

```
Creation of the initial population with proper individual dimension
Generate initial network
While not Stop Criterion do
    For (for all individuals)
        Set up corresponded NN-MVC by weight values defined in the individual
        Sample output of loop on control of NN-MVC
        Evaluate the cost (variance of sampled data)
    End for
    Set up compete among individuals and decide the winners
    Calculate mean point of winner individuals
    Reproduce new individuals using the winner of the previous generation
and information of mean point
        Consider new individuals as the new weights of NN.
End while
```

FIGURE 12. Pseudo-code for NN-MVC design by MCEP

A block scheme of the heuristic neural network process model is shown in Figure 13.

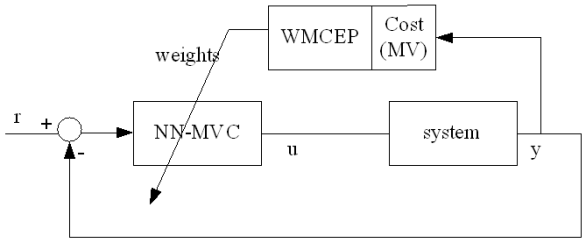

FIGURE 13. The process modelling block scheme using heuristic NN-MVC

7.1. **Simulation result.** The study example is a CSTR (Continuous Stirred-Tank Reactor) with a first-order exothermic reaction provided in [30]. It is a typical chemical engineering process which is intensively studied at the control and system identification

areas. The dynamic behavior for this CSTR (Figure 14) can be described using the following nondimensional normalized equations:

$$\dot{x}_1 = -x_1 + D_a(1 - x_1)e^{\frac{x_2}{1+x_2/\lambda}} \quad \dot{x}_2 = -x_2 + BD_a(1 - x_1)e^{\frac{x_2}{1+x_2/\lambda}} - \beta(x_2 - x_c) \tag{13}$$

where x_1 and x_2 are the reactor dimensionless concentration and temperature respectively. The case study under consideration is a regulation of outlet reactant concentration x_1. Coolant temperatures x_c is the manipulated variable. One set of parameter values $B = 1.0$, $\beta = 0.3$, $\lambda = 20.0$ and $D_a = 0.072$ which yields an open-loop system with a single stable steady state for all fixed values of the input is selected in [0 23] ([31]). The detailed nomenclature for this exothermic CSTR can be found in [32].

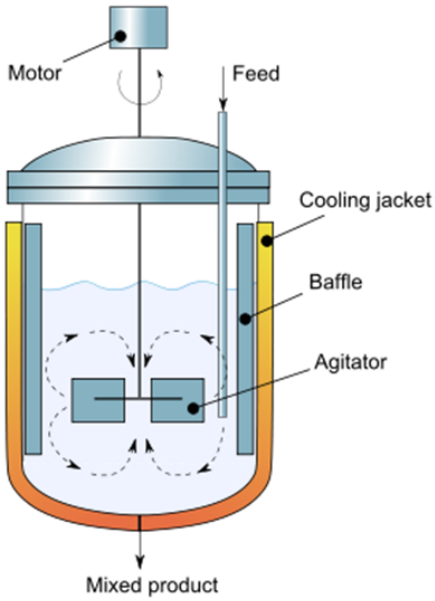

FIGURE 14. Cross-sectional diagram of continuous stirred-tank reactor

The input range real value range is [0 23]. The output is bounded at range [0 1] for introduced input range [31]. The starting situation is a stable steady situation with the initial states $x_1 = 0.6219$ and $x_2 = 3.7092$ and input $u_t = 14$.

The output Yt with an additive linear disturbance is

$$Yt = (x_1)t + Dt \tag{14}$$

where Dt is an additive disturbance. The disturbance model is an AR (AutoRegressive) model defined as:

$$Dt = \frac{e_t}{1 - 0.95q^{-1}} \tag{15}$$

where e_t is a Gaussian white noise with zero mean and variance 0.001.

NN-MVC has been designed using the WMCEP method for reducing output variance of the benchmark system. Population size for WMCEP is equal to 100; and it is repeated 20 times for reducing the effect of the chance and increasing the reliability in the results. The best results among 20 ones are included in this paper. They are run until the pre-specified generation (100) is reached. The initial population is generated randomly with variance 3. The weight values (genes) are bounded in range [−10 10]. Fitness is given by cost function J for each individual of the population, where J is the variance of output as shown in Equation (16), where the summation is performed on overall output samples

y and y_d is the desired or target value of output for a given input vector.

$$J = \sigma_y^2 = \frac{1}{N-1}\sum_{k=1}^{N}(y(k) - \bar{y} - y_d)^2, \quad \bar{y} = \frac{1}{N}\sum_{k=1}^{N} y(k) \tag{16}$$

The introduced algorithms are used to evolve the weights of the feedforward neural network with two layered structures. The input layer has eight nodes; the hidden layer has 10 hidden nodes; the output layer has *one* output nodes. Hidden transfer function is sigmoid function, and the output transfer function is a linear activation function.

Figure 15 shows the cost curve corresponding to the best cost value found in the last generation. This figure shows that the WMCEP can find the acceptable network weights in a shorter number of iterations. It admits the capability of this method in fast training the neural network MVC. The minimum variance (minimum cost) found by this the proposed algorithm is 9.37×10^{-5}.

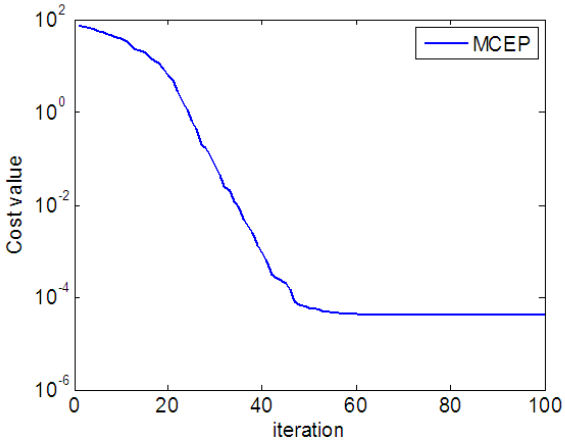

FIGURE 15. Cost values via iteration

Figure 16 shows disturbance, control signal and output data of the NN-MV designed controller by WMCEP. Optimized weights values found by WMCEP are included in Appendix A.

The optimality of the designed controller by WMCEP can be analyzed using a toolbox. The multivariate controller performance assessment toolbox was developed by the Computer Process Control Group at the University of Alberta to allow performance assessment of linear controller using the Filtering and Correlation (FCOR) Algorithm [33]. The control performance index is a single scalar usually scaled to lie within [0, 1], where values close to 0 indicate poor performance, and values close to 1 mean better/tighter control. This indeed holds when perfect control is considered as a benchmark. Here, the performance of the designed NN-MVC will be assessed by the performance assessment toolbox. The CSTR is linearized around the operating point. Then the linear model and output data of NN-MVC is applied to the performance assessment toolbox. Figure 17 shows the result of assessing the designed controller. It shows that the minimum variance index of the designed controller is 1.0281, which is higher than *one*, so the controller is better than optimal *linear* MV controller is. Moreover, this result can be admitted by the approach used in [32] for this system in the similar situation, in which linear method reaches the variance 3.51×10^{-3}, which is higher than that of the proposed method. In other words, the designed nonlinear controller has been reached to the minimum variance is lower than that of the optimal linear methods.

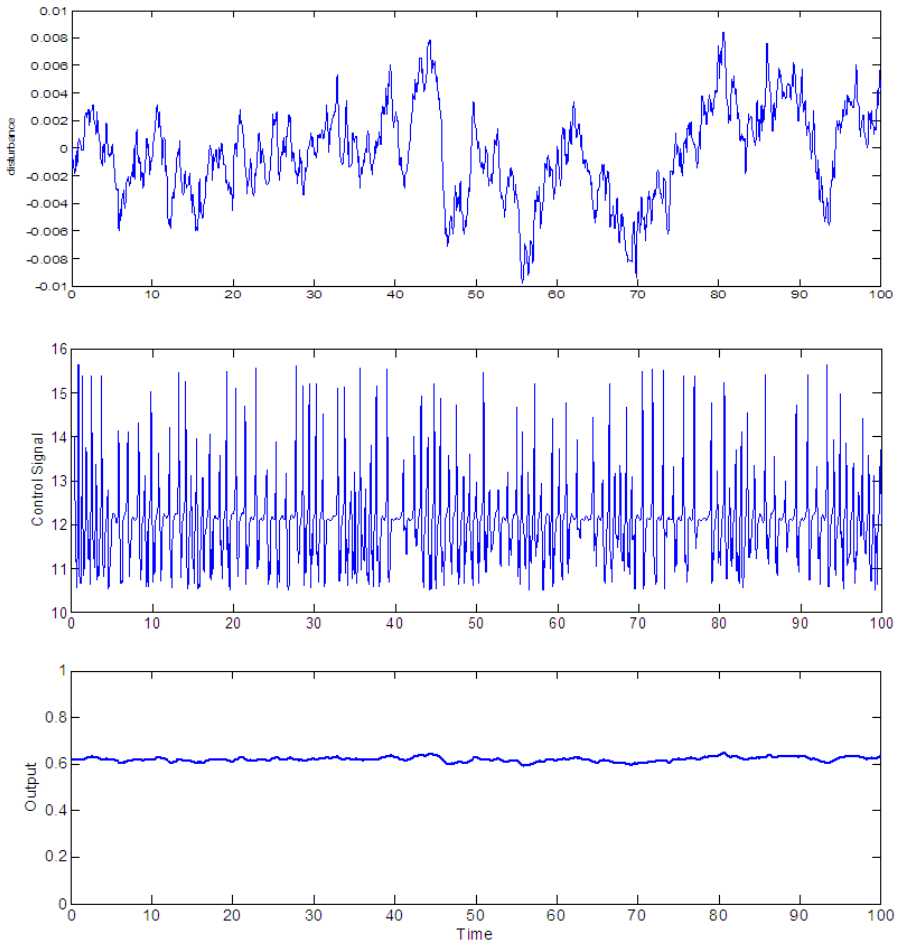

FIGURE 16. Realizations of disturbance, NN-MVC control signal and output data

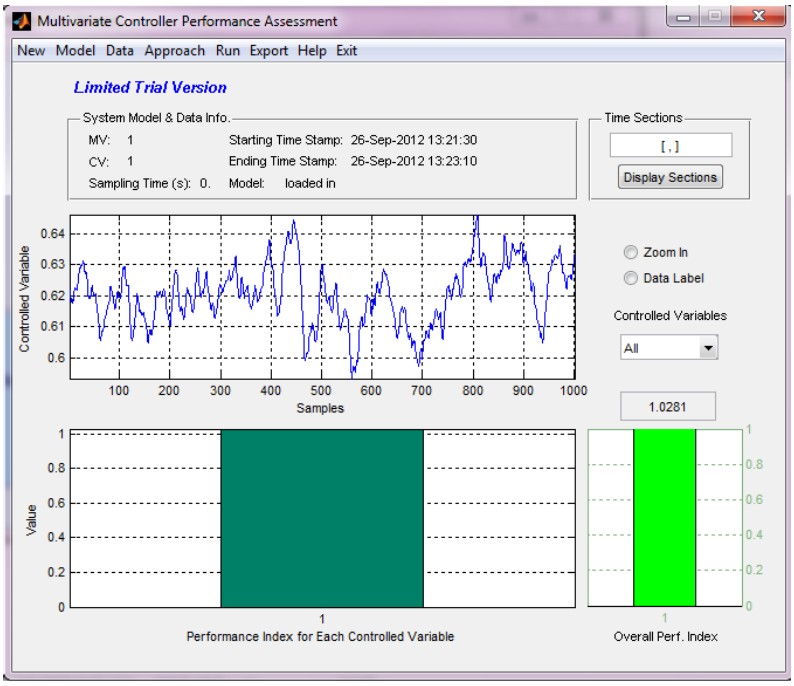

FIGURE 17. Realizations of disturbance, NN-MVC control signal and output data

8. **Conclusions.** In this paper, a developed version of CEP was proposed. There was an attempt to use the location and cost information of each parent for determining the best area for breeding the offspring. It was tried to steer the offspring toward the best parents' areas and they were prevented from getting close to worst parents using the state feedback controller. Two groups of algorithms from in and out of the EP family were selected for comparison. WMCEP had noticeable results against other EP variants. It had the best results for almost all cost functions (about 90 percent of the cost functions) in comparison with CEP, LEP, EEP, RLEP and FEP. It also demonstrated acceptable results outside the EP frontier. The proposed method was compared with eight well-known algorithms. Only CMA-ES had comparable results with WMCEP. However, CMA-ES needed heavy computation and higher CPU time for evaluating each generation. It also had problems with bounds of the variables that were not seen in the WMCEP performance. Many cost functions were selected from several references, which all of them are well-known functions in the optimization field. The proposed method is used in designing the nonlinear minimum variance controller for CSTR (Continuous Stirred-Tank Reactor) benchmark system. The designed nonlinear controller has been reached to minimum variance which is lower than that of existing methods. The future prospect of the approach proposed here is to study the accuracy of the WMCEP for obtaining better results and test it on real-world applications.

REFERENCES

[1] J. H. Holland and A. Harbor, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[2] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.

[3] I. Rechenberg, *Evolutionsstrategie: Optimierung Technischer Systeme Nach PrinzISien der Biologischen Evolution*, Frommann-Holzboog, Stuttgart, 1973.

[4] L. J. Fogel et al., *Artificial Intelligence Through Simulated Evolution*, New York, Wiley, 1966.

[5] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*, Springer, pp.2-5, 2008.

[6] Y. Alipouri, J. Poshtan, Y. Alipouri and M. R. Alipour, Momentum coefficient for promoting accuracy and convergence speed of evolutionary programming, *Applied Soft Computing*, vol.12, no.6, pp.1765-1786, 2012.

[7] N. Hansen, *The CMA Evolution Strategy: A Tutorial*, www.bionik.tu-berlin.de/user/niko/cmatutorial.pdf, 2010.

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

[9] C. Y. Lee and X. Yao, Evolutionary programming using mutations based on the levy probability distribution, *IEEE Trans. on Evolutionary Programming*, vol.8, no.1, pp.1-13, 2004.

[10] Y. Alipouri, J. Poshtan and Y. Alipouri, A modification to classical evolutionary programming by shifting strategy parameters, *Applied Intelligence*, vol.38, no.2, pp.175-192, 2013.

[11] C. T. Chen, *Linear System Theory and Design*, 3rd Edition, Oxford University Press, 1999.

[12] H. Narihisa, K. Kohmoto, T. Taniguchi, M. Ohta and K. Katayama, Evolutionary programming with only using exponential mutation, *IEEE Congress on Evolutionary Computations, Sheraton Vancouver*, Canada, 2006.

[13] H. Zhang and J. Lu, Adaptive evolutionary programming based on reinforcement learning, *Information Sciences*, vol.178, no.4, pp.971-984, 2008.

[14] K. S. Ripon, S. Kwong and K. F. Man, A real-coding jumping gene genetic algorithm (RJGGA) for multiobjective optimization, *Information Sciences*, vol.177, no.2, pp.632-654, 2007.

[15] K. S. Tang, S. Kwong and K. F. Man, A jumping gene paradigm: Theory, verification, and applications, *IEEE Circuits and Systems Magazine*, 2008.

[16] M. D. Oca, T. Stützle, M. Birattar and M. Dorigo, Frankenstein's pso: A composite particle swarm optimization algorithm, *IEEE Trans. on Evolutionary Computation*, vol.13, no.5, pp.1120-1132, 2009.

[17] *http://coco.gforge.inria.fr/doku.php?id=bbob-2009-downloads.*

[18] S. Finck, N. Hansen, R. Ros and A. Auger, Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions, *Working Paper, GECCO*, 2009.
[19] G. B. Fogel, G. W. Greenwood and K. Chellapilla, Evolutionary computation with extinction: Experiments and analysis, piscataway, *Congress on Evolutionary Computation*, USA, 2000.
[20] S. Rahnamayan, H. R. Tizhoosh and M. A. Salama, Opposition-based differential evolution, *IEEE Trans. on Evolutionary Computation*, vol.12, no.1, pp.64-79, 2008.
[21] K. V. Price, R. M. Storn and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Berlin Heidelberg, Springer, 2005.
[22] M. Montaz, C. Khompatraporn and Z. B. Zabinsky, A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems, *Journal of Global Optimization*, vol.31, no.4, pp.635-672, 2005.
[23] J. Derrac, S García, D. Molina and F. Herrera, A practical tutorial on the use of on parametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, *Swarm and Evolutionary Computation*, vol.1, pp.3-18, 2011.
[24] K. J. Astrom, *Introduction to Stochastic Control Theory*, New York, Academic Press, 1970.
[25] G. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, Holden-Day, 1970.
[26] J. Martensson, C. R. Rojas and H. Hjalmarsson, Conditions when minimum variance control is the optimal experiment for identifying a minimum variance controller, *Automatica*, vol.47, no.3, pp.578-583, 2011.
[27] M. Jelali, *Control System Performance Monitoring Assessment, Diagnosis and Improvement of Control Loop Performance in Industrial Automation*, Springer, 2010.
[28] J. Q. Gong and B. Yao, Neural network adaptive robust control of nonlinear systems in semi-strict feedback form, *Automatica*, vol.37, no.8, pp.1149-1160, 2001.
[29] D. Sbarbaro, R. M. Smith and A. Valdes, Multivariable generalized minimum variance control based on artificial neural networks and gaussian process models, *Advances in Neural Networks*, pp.52-58, 2004.
[30] F. J. Doyle, A. Packard and M. Morari, Robust controller design for a nonlinear CSTR, *Chemical Engineering Science*, vol.44, no.9, pp.1929-1947, 1989.
[31] T. D. Knapp and H. M. Budman, Robust control design of non-linear processes using empirical state affine models, *Int. J. Control*, vol.73, no.17, pp.1525-1535, 2000.
[32] W. Yu, *Variance Analysis for Nonlinear Systems*, Ph.D. Thesis, Queen's University Kingston, 2007.
[33] CPC Control Group, University of Alberta, *Multivariate Controller Performance Assessment program*, University of Alberta Computer Process Control Group, Limited Trial Version, Version 2.1, 2010.

Appendix A.

Table 7. Optimized weights values found by WMCEP

W_{ij}	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$	$j=8$	$j=9$	$j=10$
$i=1$	−2.7763	−6.1925	−1.9250	3.8635	−9.1758	6.4905	−1.2005	1.4717	−8.5236	−5.9282
$i=2$	−9.4702	−3.9335	3.8145	9.3231	−9.1604	−8.2248	3.2686	2.4480	−3.8139	7.6736
$i=3$	−2.3129	3.4643	−4.2889	−7.7741	9.0812	9.7962	10.441	−3.6376	−2.9007	3.3854
$i=4$	−8.8890	1.6644	−1.9433	−1.3899	−5.0392	−7.2601	0.4129	−2.6124	−2.4283	−0.7510
$i=5$	−2.8017	8.3229	−2.0793	4.4994	−4.6531	−5.2277	−6.6811	1.9210	1.9698	10.059
$i=6$	8.8527	5.9410	2.1878	8.4467	0.6824	0.7140	−5.3913	6.1774	1.6181	−10.700
$i=7$	3.5483	9.7259	0.0255	0.0473	3.5124	−9.5840	−7.6692	7.0161	5.6355	8.9759
$i=8$	−3.8701	6.6945	5.9696	6.2335	−1.9470	2.2898	7.3698	1.8640	−9.8845	−5.5877
W_{1j}	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$	$j=8$	$j=9$	$j=10$
$o=1$	−6.0359	4.6606	−9.6462	0.0184	3.5209	0.5573	−7.1110	−8.0571	−6.6251	−3.7482
b_j	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$	$j=8$	$j=9$	$j=10$
	8.6359	1.5325	−7.5322	3.5676	−2.8395	5.1268	−4.1412	1.5594	9.1530	4.3419
b_{11}	9.1845									

where w_{ij} is the connection weight from the ith node of input layer to the jth node of hidden layer, b_j is the threshold of the jth hidden layer input, w_{1j} is the connection weight from the jth hidden node to the output node, and b_{11} is the threshold of the output unit.