Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 709027, 9 pages
http://dx.doi.org/10.1155/2013/709027

Research Article

Improving Performance of Evolutionary Algorithms with Application to Fuzzy Control of Truck Backer-Upper System

Yousef Alipouri,[1] Saeed Ahmadizadeh,[1] Hamid Reza Karimi,[2] S. Vahid Naghavi,[3] and Ahad Soltani Sarvestani[3]

[1] *Iran University of Science and Technology, Tehran, Narmak 16846, Iran*
[2] *Department of Engineering, Faculty of Engineering and Science, University of Agder, 4898 Grimstad, Norway*
[3] *Young Researchers and Elites Club, Zarghan Branch, Islamic Azad University, Zarghan, Iran*

Correspondence should be addressed to Saeed Ahmadizadeh; ahmadizadeh.saeed5@gmail.com

Received 24 April 2013; Revised 22 September 2013; Accepted 22 September 2013

Academic Editor: Yang Xu

We propose a method to improve the performance of evolutionary algorithms (EA). The proposed approach defines operators which can modify the performance of EA, including Levy distribution function as a strategy parameters adaptation, calculating mean point for finding proper region of breeding offspring, and shifting strategy parameters to change the sequence of these parameters. Thereafter, a set of benchmark cost functions is utilized to compare the results of the proposed method with some other well-known algorithms. It is shown that the speed and accuracy of EA are increased accordingly. Finally, this method is exploited to optimize fuzzy control of truck backer-upper system.

1. Introduction

Evolutionary algorithms (EA) are usually exploited as the first candidates for hard optimization tasks. They can deal with many kinds of problems and cost functions such as multimodal, discrete, constraint on variables, high dimensionality, and stochastic cost functions; consequently, they are suitable for many applications. In the case of evolutionary computation, there are four historical paradigms that have served as the basis for much of the activity in this field, genetic algorithms (Holland, 1975) [1], genetic programming (Koza, 1992) [2], evolutionary strategies (Recheuberg, 1973) [3], and evolutionary programming (EP) [4]. The basic differences between the paradigms lie in the nature of the representation schemes, the reproduction operators, and selection methods [5].

These methods have drawn much attention to the research community in conjunction with the parallel and/or distributed computations. EP, in particular, was studied initially as a method for generating artificial intelligence [6, 7], since it is stable kind of EA and has many advantages in optimization multimodal problems. Among all proposed EP algorithms in the literature, classical EP (CEP), fast EP (FEP) [8], and Levy distributed EP (LEP) [6] are most famous variants of EP.

The classical evolutionary programming (CEP) can be presented as follows.

(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), for all $i \in \{1, \ldots, \mu\}$, where x_i's are objective variables and η_i's 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), for all $i \in \{1, \ldots, \mu\}$, 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 for $j = 1, \ldots, n$

$$x_i'(j) = x_i(j) + \eta_i(j) N_j(0,1), \quad (1)$$

$$\eta_i'(j) = \eta_i(j) \exp\left(\tau' N(0,1) + \tau N_j(0,1)\right), \quad (2)$$

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 mean zero and standard deviation 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 $(2\sqrt{n})^{-1}$.

(4) Calculate the fitness of each offspring (x_i', η_i'), for all $i \in \{1, \ldots, \mu\}$.

(5) Conduct pairwise comparison over the union of parents (x_i, η_i) and offspring (x_i', η_i'), for all $i \in \{1, \ldots, \mu\}$. For each individual, q opponents are chosen uniformly at random from all the parents and offspring. For each comparison, if the individual's fitness is no smaller than the opponent's, it receives a "win."

(6) Select the μ individuals out of (x_i, η_i) and (x_i', η_i'), for all $i \in \{1, \ldots, \mu\}$, that have the most wins to be parents of the next generation.

(7) Stop if the halting criterion is satisfied; otherwise, $k = k + 1$ and go to step (3).

Considering disadvantages arising in performance of the CEP results in searching for new methods. Although some methods have been introduced for dealing with these disadvantages, they have not attained considerable success yet. Moreover, EP family, as thought, has many advantages in dealing with multimodal cost functions, its applications in the real world are not up to mark in comparison with other evolutionary algorithms like genetic algorithm. Hence, and it is worthwhile to investigate new methods which are more successful in dealing with EP's disadvantages. Many variants of EP have been developed to boost the performance of CEP by changing (1) and (2) in step (3) [6–15]. Most attempts have introduced better distribution function in place of Gaussian mutation function such as FEP. In [8], a Cauchy-mutation-based EP, called fast EP (FEP), has been proposed, which demonstrated much better performance rather than CEP in term of converging to a near-global optimum point on some benchmark mathematical functions. FEP's success can be attributed to its greater ability to escape from local minima by using Cauchy mutation. At the same time, this will lead to difficulties in coping with some other functions. Larger jumps are beneficial when the current solution is far away from the global optimum or a better optimum, while such large jumps near the global optimum point are undesirable. In short, fast EP (FEP) is similar to CEP, but it uses a Cauchy function instead of Gaussian function mutation as the primary search operator. In [6], the authors proposed LEP (evolutionary programming using the Levy probability distribution) which can be considered as the generalization of both Gaussian and Cauchy mutation EP. Beside CEP and FEP, Narihisa et al. [9, 10] proposed EEP (exponential mutation evolutionary programming) with the mutation operator based on a double exponential probability distribution. It has been shown that the potential efficiency of EEP is compatible to the FEP. The eminent merit of EEP is that the size of search step for the solution search processes is controlled by the distribution parameter of double exponential distribution according to the convergence state for given problems [10]. Alipouri et al. [16, 17] introduced a new approach for EP family. They have shown that using some information remaining from the previous iteration of running algorithm can speed up and improve the performance of the CEP. One piece of information is average point of where parents settled in previous iteration. This information has been considered as inertia weight in (1). This method called moment coefficient EP (MCEP) changes the searching procedure of CEP through adding a new factor to produce and pull offspring toward the gathered point (mean value) of parents. In this method, gather point of individuals is assumed to be estimation of global minima.

In EP, mutation is implemented by adding strategy parameters to variable vectors of parents in order to produce offspring. When one of the strategy parameters takes on a large value, adding it to the related variable causes abrupt change. Hence, the variable grows with large steps and deviates far from the optimum point whereas some of other variables do not sense considerable changes. If this event repeats for some iteration, the variable will go further, and consequently, it slows down EP in some iteration. To avoid such an occurrence, [16] introduced a new method (shifted classical evolutionary programming (SCEP)), which can enhance the performance of classical evolutionary programming.

In this paper, speed and accuracy characterizations of EA are improved using cost and coordination information. The proposed approach defined operators which can modify performance of EA: Levy distribution function for strategy parameters adaptation, calculating mean point for finding proper region for breeding offspring, and shifting strategy parameters to change the sequence of these parameters. Thereafter, a set of benchmark cost functions are used to compare the results of the proposed method with some other known algorithms. Finally, this modified approach is exploited to optimize fuzzy control of truck backer-upper system.

The organization of this paper is as follows: Section 2 explains the EP variants; Section 3 introduces the weighted shifted Levy distributed evolutionary programming (WSLEP); and the main results are presented In Section 4. Application of the proposed method to fuzzy control of truck backer-upper system is investigated in Section 5. Finally, we conclude in Section 6.

2. Evolutionary Programming's Variants

2.1. Classical Evolutionary Programming (CEP). The general form of CEP follows a two-step process of selection and variation in a population. Following initialization of a population, the fitness of each individual in the population is scored with respect to an arbitrary fitness function. In general, selection is applied as a tournament wherein the fitness of each individual in the population is compared against the fitness of a random

set of other individuals in the same population. A "win" is recorded for an individual when individual's fitness equals or exceeds that of another in the tournament set. Individuals are then ranked with respect to the number of wins, and those with the highest number of wins over some threshold are selected as parents for the next generation. Parents are randomly varied to generate offspring, and the fitness of each member in the population is reevaluated. This process is repeated for a user-specified number of generations [18].

Offspring (x') and strategy parameters (η) are updated in each generation by (1) and (2), respectively. This process has confronted with three drawbacks.

(1) In EP, mutation is performed by adding strategy parameters to the coordinate of parents. Classical evolutionary programming uses Gaussian distribution function for updating offspring and strategy parameters (mutation parameters). Since Gaussian distribution has some drawbacks, other distribution functions are alternatively used. In the following, some well-known distribution functions will be introduced.

(2) The strategy parameters play main role in deciding the place of offspring. Determining an optimal lower bound for the strategy parameter is essential for the EP algorithm in most applications. The optimal setting of the lower bound depends on the problem and cannot be the same throughout the evolution process. In [16], shifting operator was proposed for solving this problem. In this paper, this method will be used for controlling strategy parameters step sizes.

(3) It is obvious that the basic advantages of any algorithm is in deciding the most suitable place for breeding offspring and finding the route toward the global minimum. This goal is implemented by mean point operator in MCEP [17]. In this paper, the mean point operator is used for finding optimal region for breeding new offspring.

Therefore, three methods are utilized simultaneously in order to enhance the capabilities of EP family.

2.2. Fast Evolutionary Programming (FEP). Yao et al. [8] proposed a Cauchy-mutation-based EP, called fast EP (FEP). FEP's success can be attributed to its greater ability to escape local minima by using Cauchy mutation. It demonstrates much better performance than CEP in converging to a near-global optimum point on some benchmark mathematical functions but not on all.

2.3. Levy Distributed Evolutionary Programming (LEP). LEP is a variant of EP which is similar to CEP and FEP. The difference comes from defining mutation function. LEP uses a Levy distribution function in place of Gaussian for mutation function. However, unlike FEP, in LEP only distribution function in (1) has been changed and (2) remains unchanged. So, step sizes in LEP, and CEP for strategy coefficients are similar. Practically, LEP is more similar to CEP than FEP [6].

All three distribution functions, Gaussian, Cauchy, and Levy, are special cases of the stable distributions. These distribution functions can be produced by the following equation:

$$p = \frac{x}{y^{(1/\alpha)}},$$
$$\text{if } \alpha = 1 \implies p \text{ is Cauchy random number,} \tag{3}$$
$$\text{if } 1 < \alpha < 2 \implies p \text{ is Levy random number,}$$

where x and y are independent normal Gaussian random numbers; in other words, Cauchy with zero mean and variance of 1 has the distribution of a random variable that is the ratio of two independent standard normal random variables. Similarly, Levy has distribution of a random variable that is the ratio of two independent standard normal random variables with different power as defined in (3). For example, where x is Gaussian random number with mean μ and variance σ, then $(x-\mu)^{-2}$ has a Levy distribution with location 0 and scale σ^{-2}.

3. Weighted Shifted Levy Distributed Evolutionary Programming (WSLEP)

In evolutionary computational techniques, solution of problem and the values of the individuals finally converge to a unique point. This convergence is slowly seen over the generations, and the algorithm gradually approaches the optimum point as the number of generation increases. This procedure gives us the idea of adding the average of the individuals in each generation to the algorithm to enhance the convergence speed of the EP. This technique is known as inertia weight method [17]. In order to implement this idea, first, an averaging is carried out on all of the variables. The resulted individual is called "gathered point" or "mean point." The mean value of winners in the previous iteration is regarded as the gathered point. In next step, each offspring is steered toward the mean point. For implementing this idea, coordinate of gathered point is added to coordinate of offspring. A coordinate of new individuals is calculated by adding three factors: coordinate of gathered, location of previous individuals, and strategy parameter multiplied by Levy random number. Thus, the formula of CEP can be changed for finding new individuals (1), (2) as follows:

$$x_i'(j) = a \times x_i(j) + (1-a) \times \text{mean}(j) + \eta_i(j) L_j(0,1), \tag{4}$$

$$\eta_i'(j) = \eta_i(j) \exp\left(\tau' \frac{N(0,1)}{N(0,1)^{1/3}} + \tau \frac{N(0,1)}{N(0,1)^{1/3}}\right), \tag{5}$$

where mean (j) stands for jth element of the gathered point or mean point (Figure 1), and a is defined as follows:

$$a = \frac{\text{iter}}{\text{total iter}}. \tag{6}$$

In the above equations, *total iter* is the abbreviation of the total iteration which is specified by user, and *iter* is the abbreviation of the current iteration. Figure 1 shows breeding region of offspring produced by WSLEP.

In addition, Levy distribution function is used in mutation operator presented in (4), (5) as it has long tails, thus giving offspring big step size to search map quicker which is necessary for some cost functions.

In EP, mutation is implemented by adding strategy parameters to variable vectors of parents in order to produce offspring. When one of the strategy parameters takes on a big value, adding it to the related variable causes abrupt changes in the variable. So, the variable grows with big steps and deviates far from the optimum point, whereas some of other variables do not sense considerable changes. If this event continues for some iteration, the variable will go further. This event slows down EP in some iteration. To avoid such an occurrence, [16] introduces a new method (shifting operator) that can overcome these disadvantages and enhance the performance of classical evolutionary programming. It described a modification of evolutionary programming by using a shifting method to prevent large and small changes to the strategy parameters. This method adds a function to the mutation operator. This function operates on strategy parameters and changes the sequence of these parameters.

There is a question that if the three explained approaches can enhance EP separately, whether the compound method of them (when they are used simultaneously in EP) can be more helpful or not?

Here, the LEP with weighted mean point and optimized shifting strategy parameters approach is used. The pseudo code for used method is given as follows:

Choose the initial population of individuals

Produce strategy parameters by Levy mutation function

Evaluate the cost of each individual in that population.

Repeat this generation until termination: (time limit, sufficient cost achieved, etc.)

Update the weighted mean point by (4)

Shift strategy parameters r times to the left or right (r is a random number)

Breed individuals through mutation to give birth to the offspring

Evaluate the cost of the offspring

Raise tournament to decide the next generation members

End.

4. Main Results

The main results of this paper are presented in two parts. In the first part, the seven algorithms of CEP, FEP, LEP, EEP, MCEP, SCEP, and WSLEP are compared. In this part, it will be shown that the speed and accuracy of the proposed EP improved in terms of reaching the global minimum can improve via the proposed approach. In second part, the proposed algorithm is compared with 4 algorithms: jumping gene (JG) [19], IW-PSO (Increasing Inertia Weight PSO) [20], BEA (Bacterial Evolutionary Algorithm) [21], and WSLEP from other well-known families.

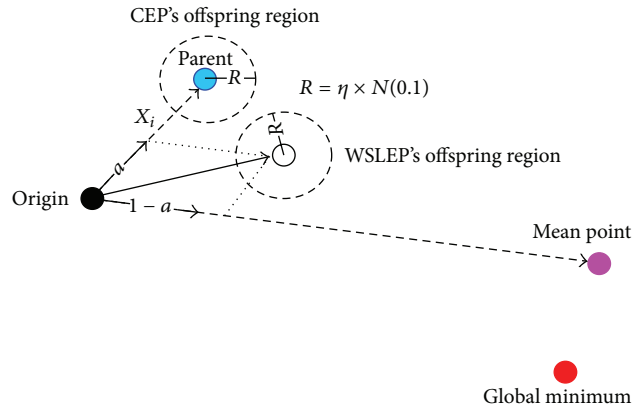

Figure 1: Location of offspring which is produced by CEP and WSLEP. Offspring is somewhere inside of the dashed circles.

Part 1. 12 well-known benchmark cost functions are selected to compare the algorithms. Descriptions of these cost functions are presented in Table 1. Some main properties of these functions will be explained, and more details can be found in [8, 22].

These cost functions can be categorized in 3 subgroups: unimodal high dimensional, multimodal high dimensional and low dimensional.

Functions f1–f4 are high dimensional-unimodal problems, which have only one global minimum that is also their only local minimum as well. Functions f5–f8 are high dimensional-multimodal functions where the number of local minima increases exponentially with the problem dimension. They are regarded as the most difficult class of problems for many optimization algorithms of which CEP has slow convergence on these functions [8]. Functions f9–f12 are low-dimensional functions, which have only a few local minima, however it is not easy to find these points by evolutionary algorithms.

The ranges of the variables and dimensions of the cost functions are chosen according to [8]. For high-dimension functions, the number of the variable is $D = 30$ (if it is not limited by the cost function). All parameters of algorithms, mentioned in Table 2, are same in all simulation results.

The best algorithm among WSLEP, CEP, FEP, EEP, MCEP, SCEP, and LEP is going to be selected on 12 test functions. To avoid any concurrence in the results, all of the algorithms have been run 20 times, and the averages of the obtained results are presented in Table 3.

All of the algorithms are run until a prespecified generation is reached. The number of generations is shown in the second column of Table 3. In this table, the best result obtained for each cost function is highlighted. These results show that the new proposed method in most cases can reach

Table 1: The 12 benchmark functions used in our experimental study; the second column introduces name of functions.

$f_1(x)$	Sphere model	$\sum_{i=1}^{n} x_i^2$
$f_2(x)$	Schwefel's problems 2.22	$\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
$f_3(x)$	Schwefel's problems 1.2	$\sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$
$f_4(x)$	Schwefel's problems 2.21	$\max_i \{ \lvert x_i \rvert, 1 \le i \le n \}$
$f_5(x)$	Generalized Schwefel's problem 2.26	$\sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert} \right)$
$f_6(x)$	Generalized Rastrigin's function	$\sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right]$
$f_7(x)$	Ackley's function	$-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos 2\pi x_i \right) + 20 + \exp(1)$
$f_8(x)$	Generalized Griewank function	$\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$
$f_9(x)$	Shekel's Foxholes function	$\left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} \left(x_i - a_{ij} \right)^6} \right]$
$f_{10}(x)$	Six-hump camel-back function	$4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$
$f_{11}(x)$	Hartman's family 2	$-\sum_{i=1}^{4} c_i \exp\left[-\sum_{i=1}^{4} a_{ij}\left(x_j - p_{ij} \right)^2 \right]$
$f_{12}(x)$	Shekel's family 3	$-\sum_{i=1}^{10} \left[(x - a_i)(x - a_i)^T + c_i \right]^{-1}$

Table 2: Parameters of the algorithms.

Tournament size q	10
Population size	100
Range bound of variables	Mentioned in last column of Table 5
Number of repetition	20
Number of generation	Mentioned in second column of Table 5

the global minimum better than other mentioned methods. All of the parameters have been considered to be the same for five methods. Furthermore, all methods have started from the same initial point (first generation). From Table 3, it can be seen that WSLEP shows the best answer for all unimodal functions. These results were expected, since WSLEP has a factor pulling offspring toward the gathered point (mean value) of parents. Unimodal cost functions have only one local minimum which is also their only global minimum. Thus, the mean value can steer offspring below the hole in a short time.

Multimodal functions make it difficult for EP families to find global minimum as they have many local and several global minimums. WSLEP has the best answer in this group too. Good results in this group are very important as there are few methods that have acceptable performance.

Part 2. In this part, four algorithms have been compared with simulation parameters cited in Table 4.

Similar to the previous part, the algorithms have been tested on 12 cost functions. The test has been repeated twenty times, and average results have been considered. Table 5 shows that the average of the best minimum has been found by algorithms in 20 time repetitions. In addition, this table shows that the accuracy of the proposed method is considerable among three other algorithms, that is, accuracy of WSLEP is better than JG, BEA, and IW-PSO.

Part 3 (*statistical test*). In recent years, use of statistical tests for evaluating 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 when 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 directly calculated from the average error results. Table 6 shows the estimations computed for each algorithm. By observing the rows of the Table 6, it can be seen that the performance of WSLEP is more suitable comparing to other algorithms, since all its related estimators are negative and it achieves low error rates of median estimation;

Table 3: Comparison algorithms on f_1–f_{12}. N is dimension of functions, F_{MIN} stands for global minimum, and last column is range of variables.

	Number of generation	CEP	FEP	EEP	LEP	SCEP	MCEP	WSLEP	N	F_{MIN}	Range bound
f_1	1500	155.67	25.194	27.34	26.685	$6.3e-6$	$2.8e-10$	**$3.5e-16$**	30	0	$[-100, 100]^n$
f_2	2000	0.418	0.596	0.746	0.707	$2.3e-4$	$7.6e-7$	**$2.5e-10$**	30	0	$[-10, 10]^n$
f_3	5000	1161	4438	2314	650	**$1.2e-8$**	$8.4e-3$	0.021	30	0	$[-100, 100]^n$
f_4	5000	0.607	34.38	2.14	1.07	0.0012	0.0002	**0.00016**	30	0	$[-100, 100]^n$
f_5	3000	−8373	**−12542**	−12154	−11167	−9500	−8230	−12162	30	−12569.5	$[-500, 500]^n$
f_6	5000	44.275	1.864	0.124	13.435	30.34	$3.1e-7$	**$3.5e-15$**	30	0	$[-5.12, 5.12]^n$
f_7	1500	8.43	3.773	0.21	5.36	0.006	0.026	**0.0016**	30	0	$[-32, 32]^n$
f_8	2000	2.373	0.971	1.03	0.694	0.009	**0.049**	**0.0049**	30	0	$[-600, 600]^n$
f_9	100	3.432	**1.047**	1.32	1.458	1.889	2.23	2.884	2	1	$[-65.53, 65.53]^n$
f_{10}	100	**−1.031**	**−1.031**	**−1.031**	**−1.031**	**−1.031**	**−1.031**	**−1.031**	2	−1.031	$[-5, 5]^n$
f_{11}	200	**−3.322**	−3.201	−3.21	−3.164	−3.30	**−3.322**	**−3.322**	6	−3.32	$[0, 1]^n$
f_{12}	100	−8.578	−8.311	−8.823	−9.145	−10.17	−9.86	**−10.52**	4	−10.5	$[0, 10]^n$

Table 4: Essential parameters of the algorithms.

General	Population size	100
	Number of repetition	50
WSLEP	Tournament size q	10
	Initial standard deviation	1
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	$\pm X_{\max}$

Table 5: Comparison algorithms on f_1–f_{12}.

	Number of generation	BEA	JG	IW-PSO	WSLEP
f_1	1500	5	$2.2e-3$	$2.1e-6$	**$3.3e-15$**
f_2	2000	0.002	$2e-6$	$2.8e-10$	**$2.4e-11$**
f_3	5000	0.68	**$7.3e-5$**	$2.4e-4$	0.021
f_4	5000	0.02	0.57	0.0074	**0.00016**
f_5	3000	−2572	−7040	−8917	−12020
f_6	5000	8.06	$1e-10$	$2.3e-7$	**$3e-15$**
f_7	1500	**$8e-5$**	$5.14e-2$	$5e-2$	0.0001
f_8	2000	0.5	**0.0022**	0.103	0.0049
f_9	100	2.03	1.74	**1.25**	2.23
f_{10}	100	−0.57	**−1.031**	**−1.031**	**−1.031**
f_{11}	200	**−3.22**	−2.86	**−3.22**	**−3.322**
f_{12}	100	−9.63	−9.95	−10.32	**−10.52**

However, FEP achieves higher error rates in this experimental study. Table 5 shows that WSLEP is most similar to MCEP in error rate (performance), as the contrast estimated method shows small difference between error rate of WSLEP and MCEP.

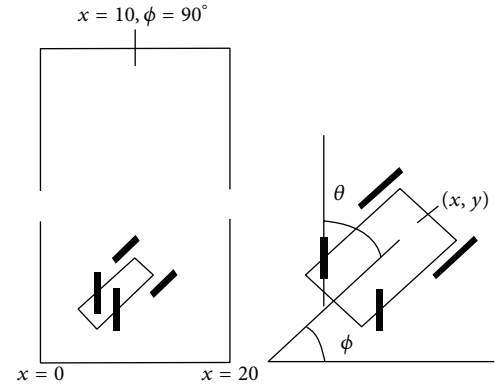

Figure 2: Simulated truck backer-upper benchmark system.

5. Fuzzy Control of Truck Backer-Upper System

In this section, the proposed EP method (WSLEP) is exploited to optimize fuzzy control of truck backer-upper system. Truck backer-upper problem is an excellent test bed for fuzzy control systems. Fuzzy controller, formulated on the basis of human understanding of the process or identified from measured control actions, can be regarded as an emulator of human operator. Controller design, however, may become difficult, especially if the numbers of state variables are large [24]. In this regard, WSLEP is used for finding the fuzzy rules, and the system is routed with a specific start point. The WSLEP algorithm uses different start points for finding optimum rules. Besides, selecting initial points and relating them in defining a cost function for learning rules are important factors. In this study, fuzzy system uses known membership function with unknown rules. Number of rules is 35. For detailed explanation, refer to [25]. The simulated truck backer-upper benchmark system is shown in Figure 2.

The truck position is determined by three state variables, φ, x, and y, where φ is the angle of the truck with respect

Table 6: Comparison of estimation method results for different algorithms.

	CEP	FEP	LEP	EEP	SCEP	MCEP	WSLEP
CEP	0	−2.23	−1.03	−1	0.57	0.58	0.59
FEP	2.23	0	1.19	1.23	2.81	2.8	2.82
LEP	1.03	−1.19	0	0.033	1.61	1.63	1.63
EEP	1	−1.23	−0.03	0	1.58	1.58	1.59
SCEP	−0.57	−2.81	−1.61	−1.58	0	0.1	0.01
MCEP	−0.58	−2.8	−1.63	−1.58	−0.1	0	$1e-4$
WSLEP	−0.59	−2.82	−1.63	−1.59	−0.01	$-1e-4$	0

Table 7: Final fuzzy rule base for the truck backer-upper control problem.

	S3	S1	S2	CE	B1	S2
	S2	CE	S3	S2	S2	S2
	S1	S2	B2	S3	S2	S2
Φ	CE	B2	B1	CE	S2	S3
	B1	S1	B1	B3	B1	CE
	B2	S1	CE	B1	CE	S1
	B3	B1	CE	B2	CE	CE
		S2	S1	CE	B1	B2
				X		

to the horizontal line. The control input to this truck is the steeling angle θ. The truck moves backward by a fixed unit distance every stage. For simplicity, we assume enough clearance between the truck and the loading dock such that y does not have to be considered as a state variable [25]. The task is to design a controller such that the system steers to final states $(x_f, \varphi_f) = (10, 90^\circ)$. We assume that $x \in [0, 20]$, $\phi \in [-90, 270]$, and $\theta \in [-40^\circ, 40^\circ]$.

For simulating purposes, we need a mathematical model of the truck. We use the following approximate model [25]:

$$\begin{aligned} x(t+1) &= x(t) + \cos\left[\phi(t) + \theta(t)\right] \\ &\quad + \sin\left[\theta(t)\right] \sin\left[\phi(t)\right], \\ y(t+1) &= y(t) + \sin\left[\phi(t) + \theta(t)\right] \\ &\quad - \sin\left[\theta(t)\right] \cos\left[\phi(t)\right], \\ \phi(t+1) &= \phi(t) - \sin^{-1}\left[\frac{2\sin(\theta(t))}{b}\right], \end{aligned} \tag{7}$$

where b is the length of the truck, and we assume that $b = 4$ in our simulations.

In step (1), we define 7 fuzzy sets in $[-90^\circ, 270]$, 5 fuzzy sets in $[0, 20]$, and 7 fuzzy sets in $[-40^\circ, 40']$, where the membership functions are shown in Figure 3.

Now, rules for fuzzy system must be designed. The cost function is

$$\text{Error} = |(x - 10)| + |\text{phi} - 90|. \tag{8}$$

The error is summed for all five initial points. The five algorithms CEP, FEP, EEP, MCEP, and SCEP along with the proposed algorithm (WSLEP) are used for minimizing the defined cost function by finding proper rules. Figure 4 shows the resulted cost via iteration curve. It can be seen that the proposed algorithm is fast and accurate to minimize the cost function (8).

The minimum founded cost is $9.4356e-004$. Table 7 shows the final fuzzy rule base for the truck backer-upper control problem.

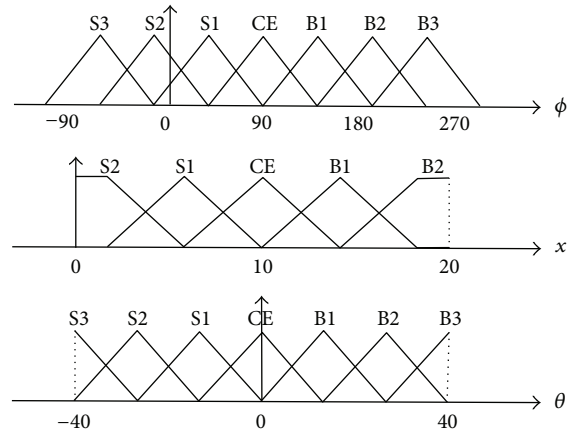

Figure 3: Membership functions.

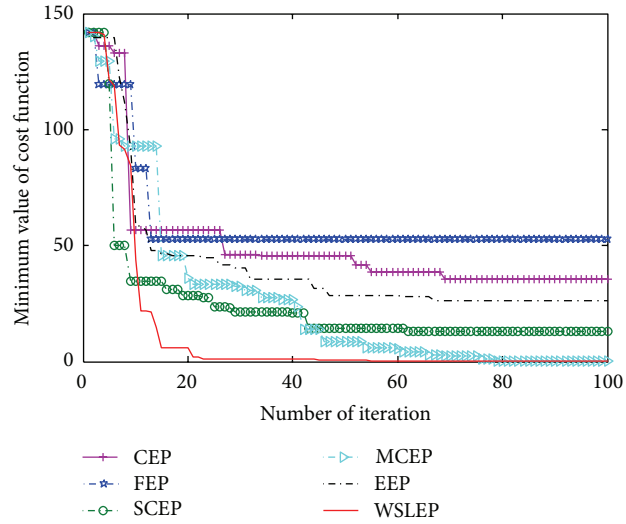

Figure 4: Cost via iteration result for optimized system using WSLEP.

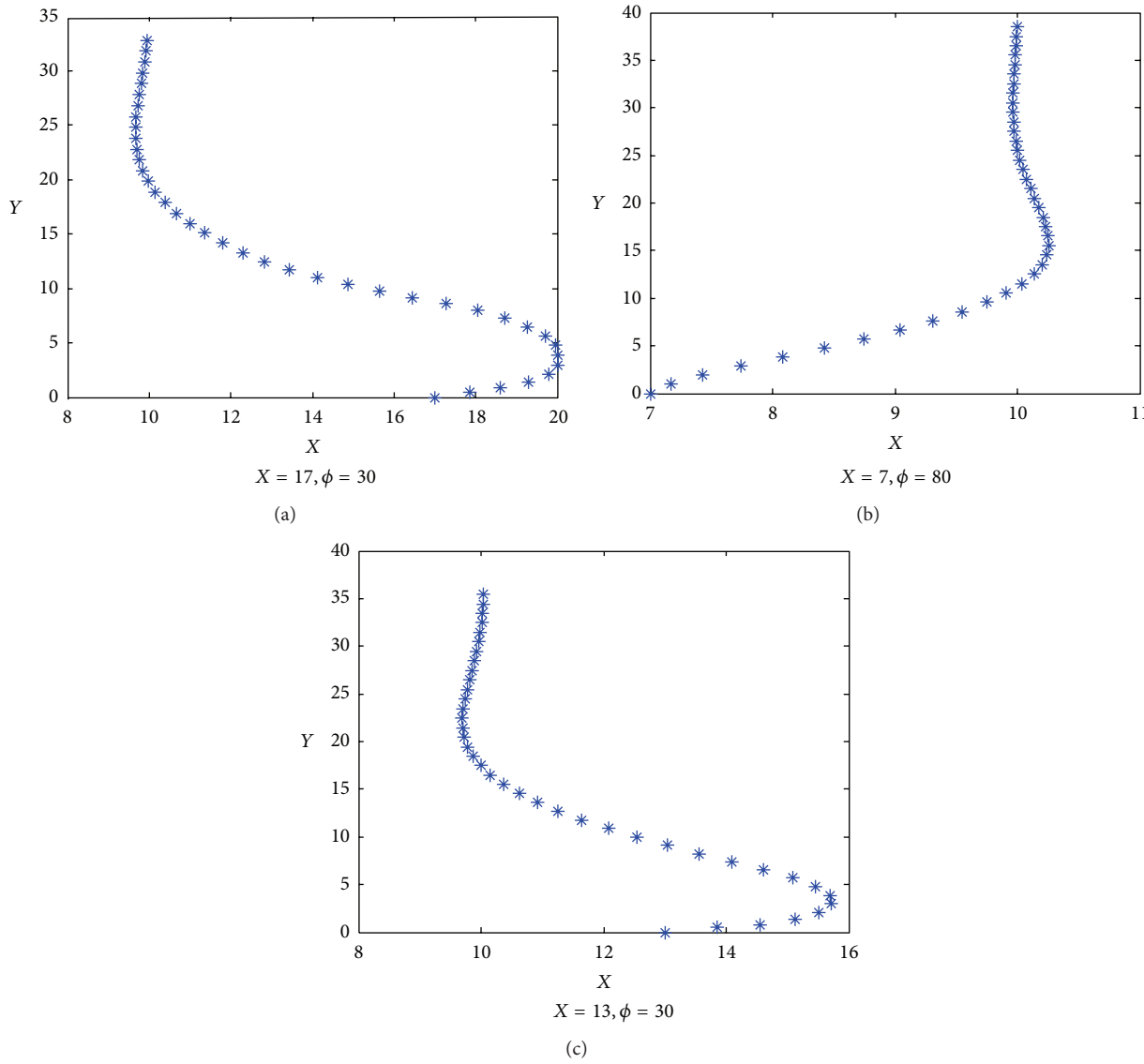

Figure 5: The truck trajectory using the designed fuzzy system as the controller for different initial conditions.

Some initial points are chosen to test the designed controller. Figure 5 shows the truck trajectory using the designed fuzzy system for different initial conditions. We see that the fuzzy controller can successfully control the truck to the desired position.

6. Conclusion

In this paper, we proposed a modified approach to increase speed and accuracy of evolutionary algorithms by designing controller using cost and coordination information. The proposed method defined operators which can improve performance of evolutionary algorithms: Levy distribution function for strategy parameters adaptation, calculating mean point for finding proper region for breeding offspring, and shifting strategy parameters to change the sequence of these parameters. Thereafter, a set of benchmark cost functions were used to compare the results of the proposed method with some other known algorithms. It was intuitively obvious that the proposed algorithm was more accurate and fast in finding the value and location of the global minimum in all three groups of the cost functions. Finally, this method was exploited to optimize fuzzy control of truck backer-upper system.

References

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.

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

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

[4] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*, Wiley, New York, NY, USA, 1966.

[5] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*, Springer, Berlin, Germany, 2008.

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

[7] X. Yao and Y. Xu, "Recent advances in evolutionary programming," *Journal of Computer Science and Technology*, vol. 3, no. 1, pp. 1–18, 2006.

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

[9] H. Narihisa, K. Kohmoto, and K. Katayama, "Evolutionary programming with double exponential probability distribution," in *Proceedings of the 2nd IASTED International Conference on Artificial Intelligence and Applications (AIA '02)*, pp. 358–363, 2002.

[10] K. Kohmoto, H. Narihisa, and K. Katayama, "Evolutionary programming using exponential mutation," in *Proceedings of the 6th World Multiconference on Systematics, Cybernetics and Informatics*, vol. 11, pp. 405–410, Orlando, Fla, USA, July 2002.

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

[12] D. B. Fogel, L. J. Fogel, and J. W. Atmar, "Meta-evolutionary programming," in *Proceedings of the 25th Asilomar Conference on Signals, Systems and Computers*, R. Chen, Ed., pp. 540–545, Maple Press, San Jose, Calif, USA, November 1991.

[13] J. W. Weibull, *Evolutionary Game Theory*, MIT Press, Cambridge, Mass, USA, 1995.

[14] X. Yao and Y. Liu, "Scaling up evolutionary programming algorithms," in *Evolutionary Programming VII*, vol. 1447 of *Lecture Notes in Computer Science*, pp. 103–112, Springer, Berlin, Germany, 1998.

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

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

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

[18] M. H. Sebt, Ya. Alipouri, and Y. Alipouri, "Solving resource-constrained project scheduling problem with evolutionary programming," *Journal of the Operational Research Society*, vol. 64, no. 9, pp. 1327–1335, 2013.

[19] W. K. S. Tang, S. T. W. Kwong, and K. F. Man, "A jumping genes paradigm: theory, verification and applications," *IEEE Circuits and Systems Magazine*, vol. 8, no. 4, pp. 18–36, 2008.

[20] A. M. M. de Oca, T. Stützle, M. Birattari, and M. Dorigo, "A composite particle swarm optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1120–1132, 2009.

[21] H.-S. Kim and P. N. Roschke, "Design of fuzzy logic controller for smart base isolation system using genetic algorithm," *Engineering Structures*, vol. 28, no. 1, pp. 84–96, 2006.

[22] J.-T. Tsai, J.-H. Chou, and W.-H. Ho, "Improved quantum-inspired evolutionary algorithm for engineering design optimization," *Mathematical Problems in Engineering*, vol. 2012, Article ID 836597, 27 pages, 2012.

[23] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[24] J. Zhang, W. Wang, Y. Zhao, and C. Cattani, "Multiobjective quantum evolutionary algorithm for the vehicle routing problem with customer satisfaction," *Mathematical Problems in Engineering*, vol. 2012, Article ID 879614, 19 pages, 2012.

[25] L. X. Wang, *A Course in Fuzzy Systems and Control*, Prentice-Hall International, London, UK, 1997.