Turkish Journal of Electrical Engineering and Computer Sciences

Volume 25 | Number 6 Article 22

1-1-2017

On the performance of newsworthy meta-heuristic algorithms based on point of view fuzzy modelling

CİHAN KARAKUZU

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2017) 25: 4706 – 4721

doi:10.3906/elk-1705-337

On the performance of newsworthy meta-heuristic algorithms based on point of view fuzzy modelling

Cihan KARAKUZU*
Department of Computer Engineering, Engineering Faculty, Bilecik Şeyh Edebali University, Bilecik, Turkey

Received: 25.05.2017 • **Accepted/Published Online:** 03.10.2017 • **Final Version:** 03.12.2017

Abstract: Heuristic algorithms are used for optimizing a large diversity of engineering problems. In this study, the performance of six commonly used and cited meta-heuristic algorithms (MhAs) is compared based on point of view fuzzy modelling. In this study, the simplest version of MhAs is specifically used to observe and evaluate their natural optimization performances. For the comparisons, dynamic system modelling using a neuro-fuzzy system (NFS) was studied. Obtained results were evaluated based on commonly used statistical metrics, and then each algorithm was ranked according to these metrics. Based on the computed average ranks, the best MhA among the six MhAs was identified, and the effect of MhAs on fuzzy modelling was investigated.

Key words: Meta-heuristic algorithms, fuzzy modelling, dynamic system, optimization

1. Introduction

It is well known that classical optimization techniques cannot solve complex engineering problems due to their solution mechanisms. Their performance depends on the solution space dimension, the number of parameters to be optimized, and the constraints on the problem. On the other hand, classical optimization algorithms have a rigid structure in terms of adapting to a given problem.

To cope with the limitations briefly mentioned above, flexible and adaptable algorithms are required to solve the problems of real life. Hence, in the literature, many meta-heuristic algorithms (MhAs) were developed and showed a superior performance to that of classical optimization algorithms. So far, MhAs have been successfully adapted to large diversity scientific problems for optimization. Following the genetic algorithm (GA), the first meta-heuristic, many MhAs have been developed, and this algorithm discovery process is unlikely to have come to an end.

The most important characteristics of MhAs are that they can solve many complex problems without complex mathematical transactions compared to classical techniques. MhAs can be considered as free derivative optimization techniques. The performance of MhAs in solving scientific or engineering problems that are very difficult and even impossible to solve with previously existing methods made them popular for about half a century. In the last few decades, there have been many studies in the literature on the use of MhAs with different problems.

Recently, the usage of MhAs for neuro-fuzzy network design has become a subject of increasing interest because of their versatile advantages such as simplicity, ease of realization, and adaptability to high-dimensional problems. Zhenget et al. [1] showed that the neuro-fuzzy network hybridized with BBO MhA shows a superior

*Correspondence: cihan.karakuzu@bilecik.edu.tr

performance to that of some other neuro-fuzzy systems (NFSs) on a classification problem. Karaboğa and Kaya [2] compared ABC algorithm training performance with GA and PSO on ANFIS based on four dynamic system modelling problems.

As seen in the literature review, MhAs have been used for solving many scientific or engineering problems. Each study in the literature has generally been linked to the success of a studied MhA, comparing a few MhAs familiar to the study's author(s) for a discussed problem, but a comparative study in the field of fuzzy system modelling using the simplest case/version of MhAs has not been found in the literature.

This paper presents a comparative performance of newsworthy MhAs on dynamic system modelling using the most preferred fuzzy network model. For the comparison particle swarm optimization (PSO), differential evolution (DE), artificial bee colony (ABC), the imperialist competitive algorithm (ICA), biogeography-based optimization (BBO), and the cuckoo optimization algorithm (COA) have been chosen with regards to the number of citations they have received and their popularity in the scientific community. Specifically, the simplest case/version of the MhAs was used to compare their performance-based searching or optimizing nature. The contributions of this paper are as follows: (i) being the first to carry out a comparative study of newsworthy meta-heuristics for fuzzy modelling on five selected nonlinear systems, (ii) being the first study to consider the performance of the simplest case MhAs using a novel well-rounded comparison systematic, (iii) identifying which MhA is the best among the six, and (iv) investigating the effect of MhAs on fuzzy modelling.

2. Newsworthy meta-heuristic algorithms

The algorithms used in this study are introduced briefly in this section.

2.1. The particle swarm optimization (PSO) algorithm

PSO was inspired by social interactions among animals like the flocking of birds. PSO searches for the optimum location in the solution space by means of its individuals called particles. The used case of PSO is formulated by Eqs. (1) and (2) [3]. In the equations, r_1 and r_2, and c_1 and c_2, and ω are uniformly distributed random numbers, cognitive/social constant learning rates, and weighting factor, respectively. The velocity given in Eq. (1) is limited within the range of $[\mathrm{v}_{min}, \mathrm{v}_{\max}]$. Particles are updated by Eq. (2) at each generation.

$$v_i(n+1) = \omega v_i(n) + c_1 r_1 (l_{best,i} - p_i(n)) + c_2 r_2 (g_{best} - p_i(n)) \tag{1}$$

$$p_i(n+1) = p_i(n) + v_i(n+1) \tag{2}$$

2.2. The differential evolution (DE) algorithm

DE is a popular and commonly used MhA algorithm discovered by Storn and Price in 1996 [4]. The most important feature of DE is that it generates trial parameter vectors using operations like those of GA [5]. DE uses the difference between two individuals randomly selected from a population to add a third individual. The DE algorithm used in this study is given in Algorithm 1.

2.3. The artificial bee colony (ABC) algorithm

ABC is a basic and efficient MhA introduced by Karaboğa in 2005 inspired by the nectar exploration behavior of a honey bee swarm [6]. The detailed pseudo-code of the ABC algorithm can be obtained from [7].

Algorithm 1 The DE algorithm in pseudo-code

Generate the initial population (N≥4) and set algorithm parameters {mutation scale factor F ∈ [0, 2], crossover rate CR∈ (0,1)}

Do (Main loop)

For each individual x_i in the population

Choose randomly three numbers $\{r_1, r_2, r_3 \in [1, N]$ and $r_1 \neq r_2 \neq r_3 \neq i\}$

Generate a random integer $j_{rand} \in [1, D]$

For each element of x_i

$$u_{i,j} = \begin{cases} x_{r1,j} + F\,(x_{r2,j} - x_{r3,j}) & if\,(rand \leq CR \vee \ j = j_{rand}) \\ x_{i,j} & otherwise \end{cases}$$

End For

Replace x_i with the offspring u_i if u_i is better

End For

Until the termination condition is achieved

Solution: The individual having the best fitness will be the solution for the problem

2.4. The imperialist competitive algorithm (ICA)

ICA is a MhA discovered by Atashpaz-Gargari and Lucas in 2007 [8]. Its source of inspiration is the world system of imperialist countries. The basic steps of ICA used in this study are given in Algorithm 2.

2.5. The biogeography-based optimization (BBO) algorithm

The BBO algorithm is a relatively new MhA based on a biogeography concept. Its source of inspiration is the idea of the migration strategy of animals. BBO was proposed by Dan Simon in 2008 [9]. The pseudo-code of BBO used in this study is given in Algorithm 3.

2.6. The cuckoo optimization algorithm (COA)

The COA is the newest MhA used in this study. It was proposed by Ramin Rajabioun in [10]. Its source of inspiration is the lifestyle of the cuckoo bird. The extraordinary egg laying and breeding behaviors of cuckoos constitute the main operations of this new metaheuristic optimization algorithm. The code of the COA given in (https://www.mathworks.com/matlabcentral/fileexchange/35635-cuckoo-optimization-algorithm) was adapted to the problems studied here and used.

3. Dynamic systems and comparison bench

3.1. Benchmark dynamic systems

In this section, benchmark dynamic systems (BDSs) used to investigate performance of MhAs, as briefly defined in the previous section, are introduced. The systems are listed in Table 1. These systems are often used in the

Algorithm 2 The ICA algorithm in pseudo-code

1. **Initialization and setting algorithm parameters**

   (a) **Set** the number of countries (N) and imperial countries (N_{imp}), searching space dimension (D), revolution rate (r_v), assimilation coefficient $\beta \in [0,1]$, total cost coefficient ξ, uniting threshold δ, maximum generation number (G_{max}), termination criterion (TC)

   (b) **Assign** initial countries in the D dimensional search space (c_i, i=1, 2, 3, . . .N))

   (c) Evaluate the fitness f(c_i) of each country

   (d) Determine the initial empires countries

2. **Do (**Main loop**)**

   (a) For each empire i **do**

      a) Perform **assimilation** for all colonies in empire i

      b) Perform **revolving** for all colonies in empire

      c) **Evaluate** the fitness for all colonies in empire i

      d) Exchange positions of the imperialist and a colony of empire i according to fitness values

      e) Compute of total cost for empire i

      $$\text{TotalCost}_i = \text{ImperialistCost}_i + \xi \ \text{mean}_i(\text{ColoniesCosts})$$

   (b) Uniting the empires if distance between each other is less than uniting threshold δ

   (c) Perform imperialist competition

      Eliminating the powerless empires

   Until the termination condition is achieved

3. **Solution:** The imperialist having the best fitness will be the solution for the problem

literature as benchmarks for different modelling purposes. In the table, the used input arrays of the systems for both training and test stages are defined as well as their mathematical expressions.

3.2. Neuro-fuzzy network structure

In this study, a Takagi–Sugeno (TS)-type fuzzy inference system in a five-layer neural network structure was used as a comparison bench. Since it combines the advantages of fuzzy logic and neural network techniques in

Algorithm 3 The BBO algorithm in pseudo-code

1. **Initialization and setting algorithm parameters**

   (a) **Set** the population size (N), searching space dimension (D), number of elite (n_E), mutation probability (m), maximum generation number (G_{max}), termination criterion (TC)

   (b) **Assign** initial population members (x_i) in the D dimensional search space

   (c) Evaluate the fitness of each population member { $f(x_i)$ }

   (d) **Sort** the population and save the best n_E solutions in the elite array

   (e) **Assign** a zero-temporary population (z)

   (f) Compute migration rates λ, assuming the population is sorted

$$\mu = [\frac{N}{N+1}\frac{N-1}{N+1}\frac{N-2}{N+1}\cdots\frac{1}{N+1}], \lambda = 1-\mu$$

2. **Do (**Main loop**)**

   (a) For each member i **do**

   Perform **emigration** (roulette wheel selection) kth dimension

   Emigration Yes: $z_{i,k} = x_{selected,k}$ Emigration No: $z_{i,k} = x_{i,k}$

   (b) Perform **mutation** for all members in temporary population

   (c) **Replace** members with their new migrated and mutated versions {x = z}

   (d) **Sort** the population from best to worst

   (e) Replace the worst individuals with the previous generation's elites

   (f) **Sort** again the population from best to worst

   (g) **Save** the best member as solution of the current generation

   Until the termination condition is achieved

3. **Solution:** The best member having the best fitness will be the solution for the problem.

a single network structure, it could be called NFS. For the efficient and optimal usage of NFS, MhAs are used for parameter tuning [15]. NFS, as a bench, was used to model nonlinear systems for performance comparison of MhAs here. NFS parameters are optimized by MhAs for system modelling purposes.

Table 1. Benchmark dynamic systems used for system modelling based on fuzzy logic learned by MhAs.

#	BDSs	Train	Test
1	$y(k)=\frac{y(k-1)y(k-2)(y(k-1)+2.5)}{1+y^2(k-1)+y^2(k-2)}+u(k)$[11]	$u(k)=\cos\frac{2\pi k}{100}$	$u(k)=\sin\frac{2\pi k}{25}$
2	$y(k+1)=\frac{y(k)}{1+y^2(k)}+u^3(k)$[11]	$u(k)=\cos\frac{2\pi k}{100}$	$u(k)=\sin\frac{2\pi k}{25}$
3	$y(k+1)=y(k)+u(k)e^{-3\lvert y(k)\rvert}$ [12]	Random amplitude in interval of [-1 1] with 10 sampling periods	
4	$y(k+1)=\frac{24+y(k)}{30}y(k)-0.8\frac{u(k)^2}{1+u(k)^2}y(k-1)+0.5u(k)$[13]	Random amplitude in interval of [-5 5] and random sampling periods in interval of [1 10]	
5	$y(k+1)=0.5(\frac{y(k)}{1+y^2(k)}+(1+u(k))u(k)(1-u(k)))$[14]	Random amplitude in interval of [-2 2] and random sampling periods in interval of [1 10]	

A fuzzy system having R first order TS rules in the rule base, Gaussian membership functions (MFs), product inference engine, and weighted average defuzzifier was used in this study. The input–output relation of a NFS with two inputs (x, y) and a single output (z) is given by (3), where p_l, q_l, and r_l are the lth rule's consequent parameters and μ_{li} is the lth rule's ith membership function.

$$z(x,y)=\frac{\sum_{l=1}^{R}(p_lx+q_ly+r_l)\mu_{li}(x)\mu_{lj}(y)}{\sum_{l=1}^{R}\mu_{li}(x)\mu_{lj}(y)} \tag{3}$$

The Gaussian membership function is given by (4), where c_{lj} and σ_{lj} are the center and variance of the related Gaussian membership function. Comprehensive information on the TS-type fuzzy system with a five-layer network structure can be acquired from [16].

$$\mu_{li}(x)=\exp\left(-\frac{1}{2}\left(\frac{x-c_{li}}{\sigma_{li}}\right)^2\right),\ \mu_{lj}(y)=\exp\left(-\frac{1}{2}\left(\frac{y-c_{lj}}{\sigma_{lj}}\right)^2\right) \tag{4}$$

To model BDS, NFSs were structured as shown in Table 2. The modelling or optimizing process can be called training. During the training stage, the parameters of NFS were optimized by a meta-heuristic algorithm. After this stage, the NFS model of each DBS was obtained. The performances of these models were observed by using test data that were invisible at the training stage. In both stages, the two data sets (training and test), were prepared beforehand. The used input arrays to obtain these sets are graphically given in Figures 1 and 2, respectively. Figure 3 summarizes the training stage.

4. Comparison of MhAs

For comparison purposes, each MhA was separately run 50 times with 100 generations for each BDS to model it via the NFS network. Each run included, first, a 100-generation training phase using a training set and, second,

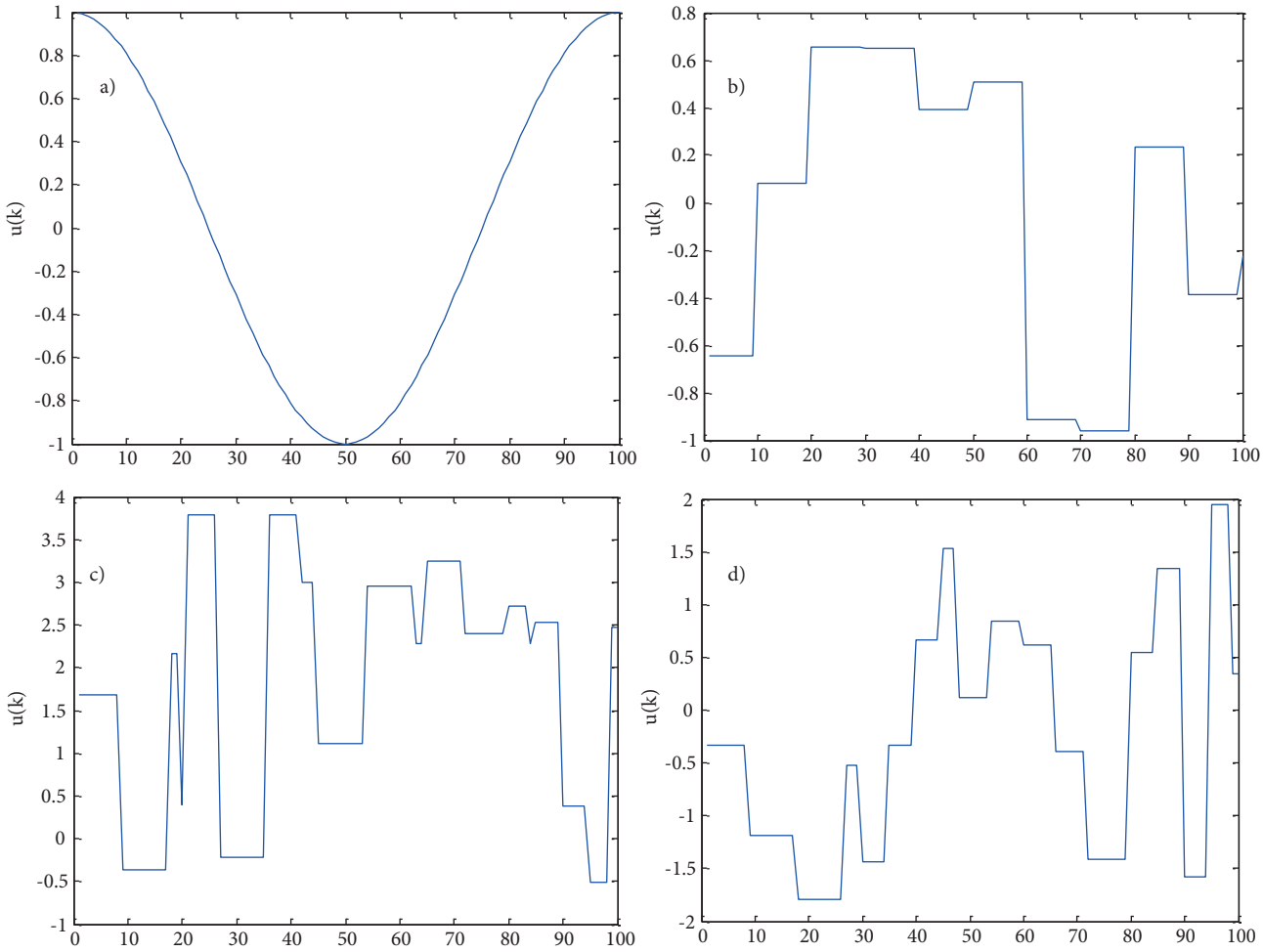

Figure 1. Input arrays (u(k)) to obtain training sets: (a) BDS1 and BDS2, (b) BDS3, (c) BDS4, (d) BDS5.

Table 2. NFS structures to modelling BDSs.

#	Inputs	# of MFs	# of rules	# of parameters (D)
BDS1	u(k), y(k-2), y(k-1)	2, 2, 2	8	36
BDS2	u(k), y(k), y(k-1)	2, 2, 2	8	36
BDS3	u(k), y(k)	2, 2	4	20
BDS4	u(k), y(k), y(k-1)	2, 2, 2	8	36
BDS5	u(k), y(k)	2, 2	4	20

a testing phase using a test set. The fitness value of an individual of a running algorithm, the performance of the solution at the end of the training phase, and then the performance of the solution for the testing phase were measured by the following MSE metric:

$$cost = \frac{1}{k_{\max}} \sum_{k=1}^{k_{\max}} e\left(k\right)^2 \tag{5}$$

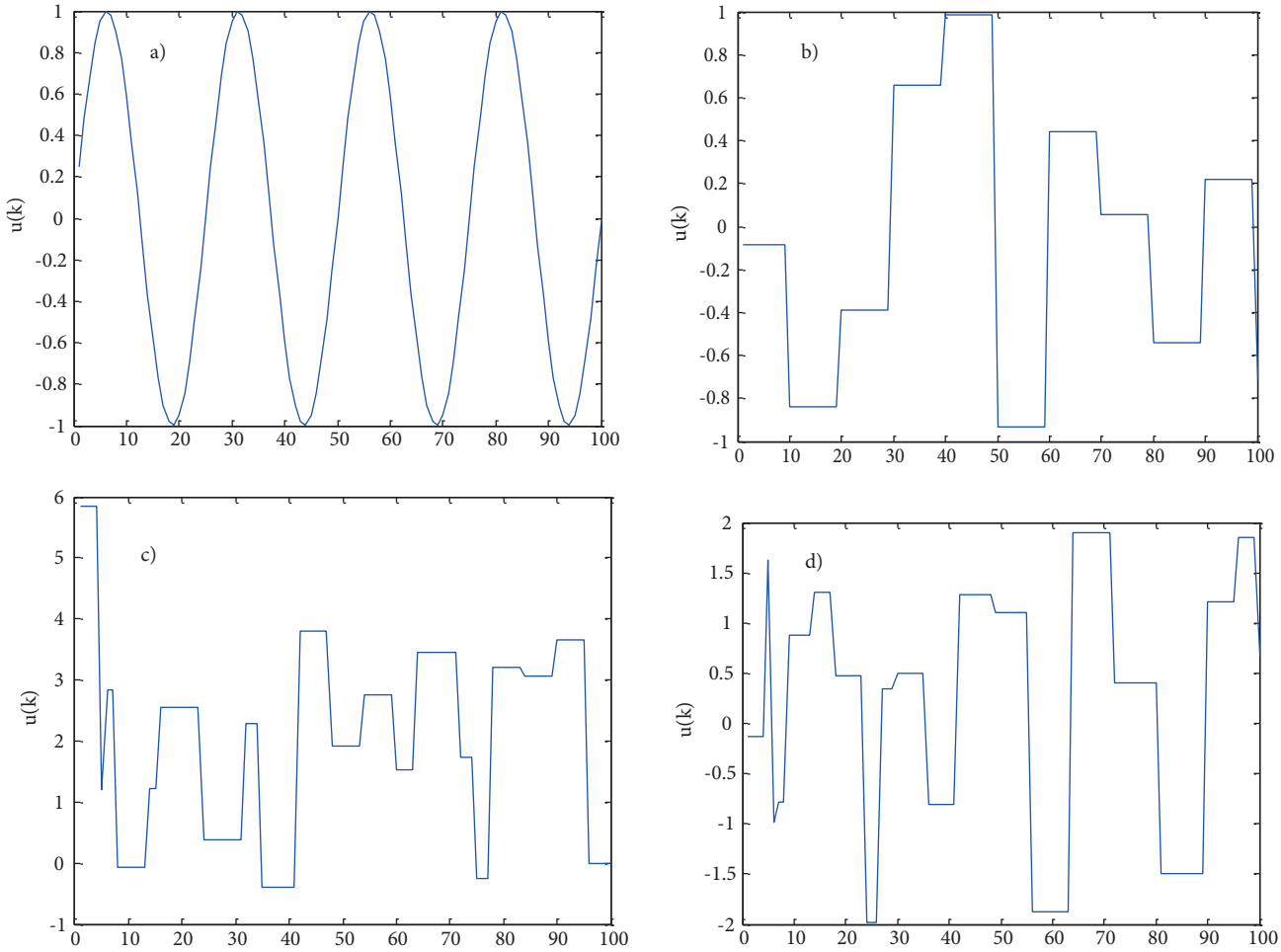

Figure 2. Input arrays (u(k)) to obtain testing sets: (a) BDS1 and BDS2, (b) BDS3, (c) BDS4, (d) BDS5.

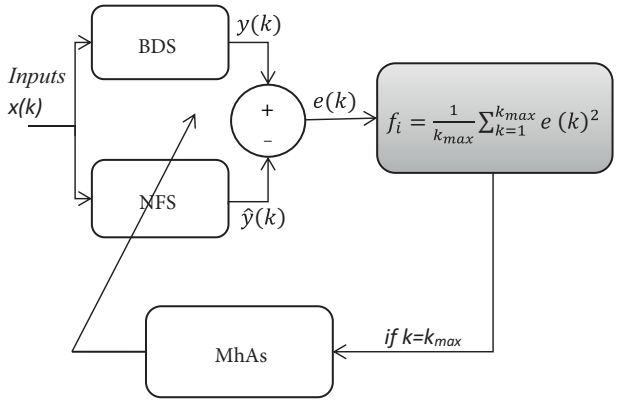

Figure 3. BDS modelling block structure using NFS by MhAs.

The parameters of each MhA are given in Table 3. These parameter values were commonly used or defined by the founder of the related algorithm. The simplest version of MhAs was used to observe and evaluate the natural optimization performances of the algorithms in this study. For all algorithms, the swarm/population/colony size was used as defined in Eq. (6) taken from [17].

Table 3. Parameters of MhAs

	Metaheuristics					
	PSO	DE	ABC	ICA	BBO	COA
Parameters*	ω=0.8	CR =0.4	$L = N\frac{D}{2}$	N_{imp}=4	m=0.04	$C_{initial}$=4
	c_1=2.05	F =0.3		r_v=0.3	n_E=5	$\min_{Egg}$=2
	c_2=2.05			β=2		$\max_{Egg}$=4
				ξ=0.02		r_{egg}=0.01
				δ=0.02		$n_{cluster}$=2
						λ=2

* The definition of each parameter can be found in the related pseudo-code or reference given in Section 2.

$$N = round(10 + 2\sqrt{D}) \tag{6}$$

The obtained results of 50 runs of each MhA for modelling each BDS were considered according to three statistic metrics: average cost value, average standard deviation, and average elapsed time. In Table 4, the average cost values obtained at the end of the runs are given for both the training and the testing phases. As can be seen from the table, for each BDS, ranks of MhAs were determined based on their average cost values in the column labelled Rnk. The average ranks of each MhA were computed based on these rank values. Table 5 gives the statistical results of the standard deviation of the average cost values of the runs in a manner similar to that of Table 4. Similarly, Table 6 summarizes the result of the average elapsed time and the standard deviation for the runs.

To evaluate convergence in terms of a graphical comparison, the point of view changes of the average cost values of the best individuals for all runs during the course of the training stage are given in the left column of Figure 4. The column on the right in the figure shows the course of the training stages of BDS modelling with NFS for the best run among all the runs. From the graphics in Figure 4, the convergence behavior of each MhA can be visually evaluated.

To give a general outline to the readers, Figure 5 gives the graphical performances of the solutions obtained at the end of the best run for each BDS. As can be seen from the figure, a training performance with 100 generations was adequate for BDS1, BDS2, BDS3, and BDS5 but not for BDS4. The input design or NFS structure may not be appropriate for BDS4. One should not dwell on these deficiencies since the main focus is not this kind of modelling, but general learning performance of MhAs. On the other hand, except BDS3, it can be stated that testing performances were generally on a parallel course with training performances. It must be noted that these results were for only 100 generations. The observed lack of modelling may not be seen for larger generations such as 500 or 1000.

5. Conclusions and remarks

The main objective of this study was to observe and make conclusions about the optimization performance of commonly used and cited newsworthy MhAs on fuzzy modeling problems. MhAs were run to tune parameters of NFS for the modelling of benchmark dynamic systems using the simplest version of each MhA to observe and evaluate their natural optimization performances in this study. Results obtained by the repeated running of MhAs on the problems were concluded primarily in four measuring metrics.

The first metric is the average cost values obtained at the end of the runs. Table 4 shows the values of this metric for both the training and testing phases of each BDS. In the table, the ranks and average ranks

Table 4. Average cost values of 50 runs with 100 generations.

	BDS#	Metaheuristics											
		PSO		DE		ABC		ICA		BBO		COA	
		Cost	Rnk	Cost	Rnk	Cost	Rnk	Cost	Rnk	Cost	Rnk	Cost	Rnk
Training	BDS1	0.089817	3	0.127462	4	0.188333	5	0.311674	6	0.064901	2	0.061064	1
	BDS2	0.034225	2	0.039679	4	0.047525	5	0.093878	6	0.024713	1	0.036817	3
	BDS3	0.566246	6	0.06198	2	0.124549	3	0.13077	4	0.142918	5	0.022967	1
	BDS4	0.145498	3	0.132029	2	0.268095	5	0.285517	6	0.092052	1	0.148646	4
	BDS5	5.567664	6	0.782666	3	1.237623	5	0.849726	4	0.699079	2	0.44426	1
Testing	BDS1	0.322227	4	0.286016	3	0.507851	5	0.707768	6	0.209454	1	0.233808	2
	BDS2	0.12885	3	0.100238	1	0.14483	4	0.255404	6	0.109411	2	0.202667	5
	BDS3	8.396881	6	2.751276	2	3.7859	3	4.567569	4	4.80249	5	2.172251	1
	BDS4	0.363079	1	0.659749	2	2.094175	5	2.298916	6	0.847537	3	1.390079	4
	BDS5	3.817961	6	0.652551	4	1.018843	5	0.57094	2	0.587524	3	0.372261	1
Average rank			4		2.7		4.5		5		2.5		2.3

Table 5. Standard deviation of cost values obtained for training and test data sets.

	BDS#	Metaheuristics											
		PSO		DE		ABC		ICA		BBO		COA	
		Std. Dev.	Rnk	Std. Dev.	Rnk	Std. Dev.	Rnk	Std. Dev.	Rnk	Std. Dev.	Rnk	Std. Dev.	Rnk
Training	BDS1	0.065330	4	0.064043	3	0.084921	5	0.235195	6	0.032983	1	0.033292	2
	BDS2	0.040696	5	0.018881	1	0.026279	3	0.063761	6	0.020076	2	0.036926	4
	BDS3	0.718248	6	0.049960	2	0.097349	3	0.176797	5	0.167026	4	0.022870	1
	BDS4	0.142775	5	0.064865	2	0.113058	3	0.154041	6	0.038572	1	0.130752	4
	BDS5	6.191930	6	0.435738	4	0.645328	5	0.386051	2	0.425510	3	0.290821	1
Testing	BDS1	0.040696	1	0.164445	2	0.350352	5	0.722711	6	0.173132	3	0.198535	4
	BDS2	0.099444	4	0.072974	2	0.026279	1	0.205327	5	0.095677	3	0.249947	6
	BDS3	9.786736	5	4.351444	1	6.825202	4	16.44516	6	5.865152	2	6.252612	3
	BDS4	0.439595	1	0.676739	2	4.302287	5	5.068684	6	0.999580	3	3.831684	4
	BDS5	5.655163	6	0.287485	4	0.532917	5	0.268291	3	0.261945	2	0.233534	1
Average rank			4.3		2.3		3.9		5.1		2.4		3

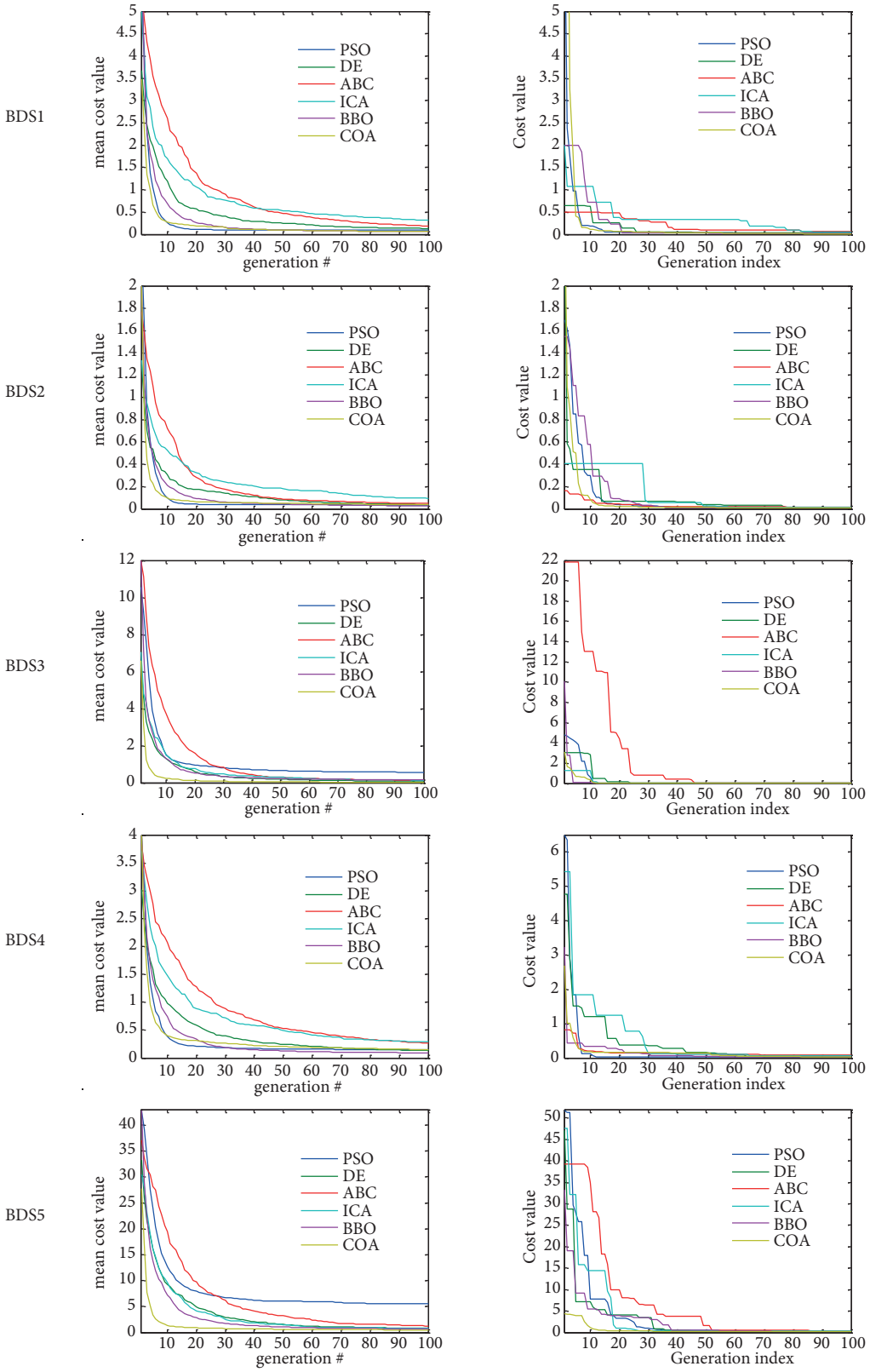

Figure 4. Training course of MhAs in terms of mean cost for all runs (left column) and in terms of cost for the best run (right column).

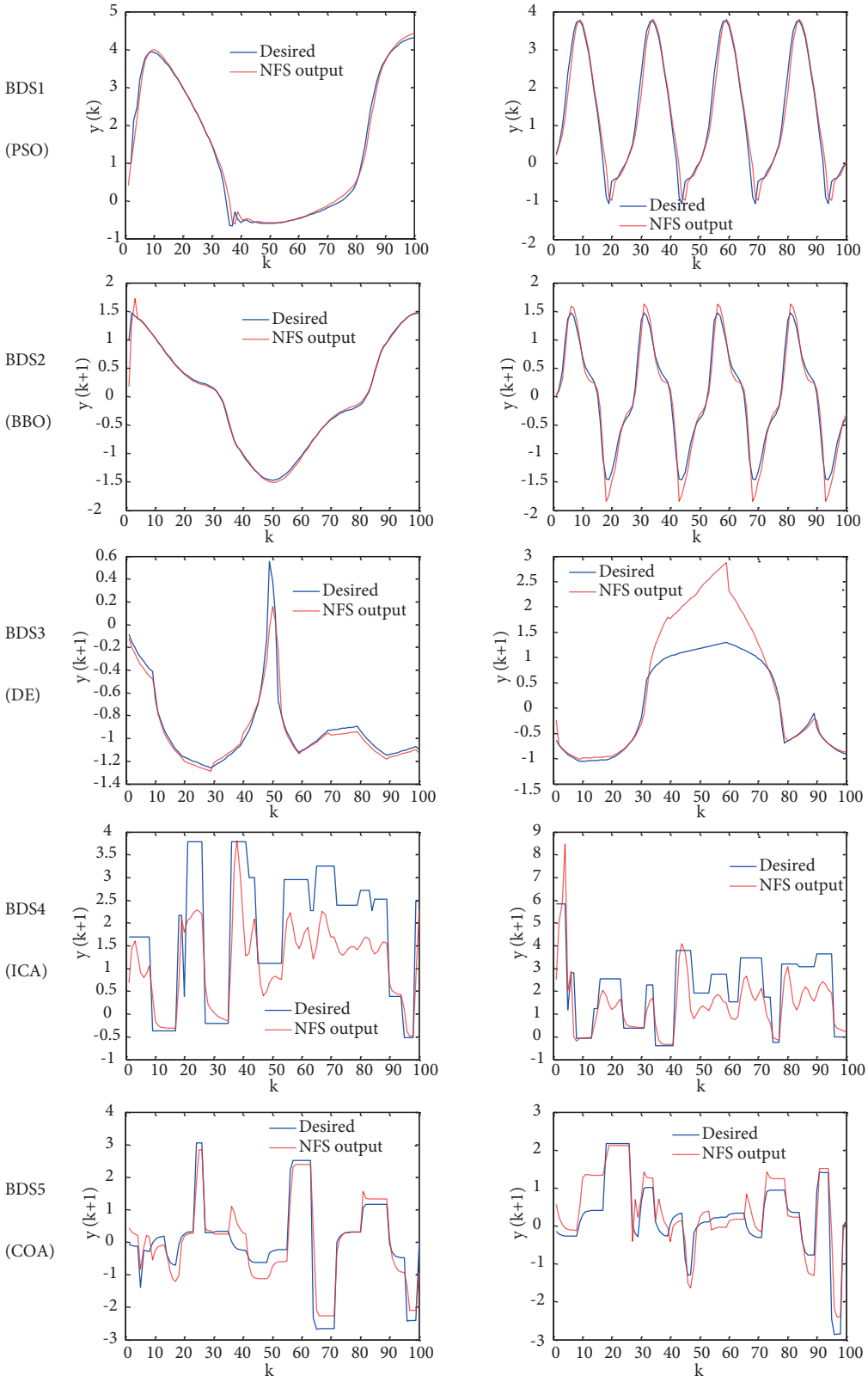

Figure 5. Training (left column) and testing (right column) performance of the solution obtained at the end of the best run among all runs with 100 generations.

Table 6. Average elapsed times of runs and their standard deviation obtained for the training.

	BDS#	Metaheuristics					
		PSO	DE	ABC	ICA	BBO	COA
		Time [s] Rnk	Time [s] Rnk	Time [s] Rnk	Time [s] Rnk	Time [s] Rnk	Time [s] Rnk
Mean elapsed time	BDS1	2.042792 3	2.085862 4	2.089168 5	1.200190 1	1.310110 2	252.5919 6
	BDS2	2.114711 4	2.065126 3	2.155981 5	1.126964 1	1.183497 2	244.3371 6
	BDS3	1.375803 2	1.405225 3	1.490951 5	1.309137 1	1.481025 4	262.6119 6
	BDS4	2.104275 3	2.136721 4	2.251098 5	1.732459 1	2.041052 2	346.6998 6
	BDS5	1.441667 4	1.386875 3	1.472106 5	0.806271 2	0.700518 1	174.2024 6
Standard deviation	BDS1	0.000409 3	0.000417 4	0.000418 5	0.000240 1	0.000262 2	0.050518 6
	BDS2	0.000423 4	0.000413 3	0.000431 5	0.000225 1	0.000237 2	0.048867 6
	BDS3	0.000275 2	0.000281 3	0.000298 5	0.000262 1	0.000296 4	0.052522 6
	BDS4	0.000421 3	0.000427 4	0.000450 5	0.000346 1	0.000408 2	0.069340 6
	BDS5	0.000288 4	0.000277 3	0.000294 5	0.000161 2	0.000140 1	0.034840 6
Average rank		3.2	3.4	5	1.2	2.2	6

of each algorithm are given. According to this metric, COA is the best MhA, but the average ranks of BBO and ICA are close to that of COA. The others' average ranks are far from the best ones. The second metric is the average standard deviation of cost values obtained in the last generation of runs. Thus, if the standard deviation value decreases, it means that the algorithm finds an approximately identical solution. On the other hand, if it increases, the algorithm finds different solutions for each run. The standard deviations of the cost values of 50 runs are given in Table 5. As seen from the table, DE is the best MhA, and the next best is BBO, according to the average ranks.

The average elapsed time of 50 runs is the third statistical metric used in this study. The numerical results for this metric are given together with its standard deviation in Table 6. This table presents remarkable comparison results. According to ranks values given in the table, COA is, surprisingly, the slowest one while ICA is the fastest algorithm among the MhAs. The others are sorted as BBO, PSO, DE, and ABC from good to bad. It was concluded that the slowness of COA arises particularly from the clustering transaction beside its other operations like egg laying, removing eggs, sorting, and combining habitat. On the other hand, ICA is the fastest of them although it has similar complex operations such as assimilation and revolving to that of COA except clustering. The last metric is convergence speed. The convergence behavior of MhAs cannot be easily seen in the tables but is possible to understand from the graphics given in Figure 4. From the figure, the convergence speed of each algorithm can be observed. Table 7 shows the convergence speed ranks of MhAs

considering their convergence in the first decade's generations. According to the average ranks given in the table, COA is the fastest converging MhA, and BBO and PSO can be considered the second fastest since they have ranks close to each other; the others have similar converging performances.

Table 7. Convergence speed comparison table from Figure 4.

	BDS#	Metaheuristics					
		PSO Rnk	DE Rnk	ABC Rnk	ICA Rnk	BBO Rnk	COA Rnk
For average of all runs	BDS1	1	4	6	5	3	2
	BDS2	2	4	5	6	3	1
	BDS3	6	3	5	4	2	1
	BDS4	1	4	6	5	3	2
	BDS5	5	4	6	3	2	1
For the best run	BDS1	2	4	5	6	3	1
	BDS2	3	4	2	6	5	1
	BDS3	3	5	6	4	1	2
	BDS4	1	5	3	6	4	2
	BDS5	4	3	6	5	2	1
Average rank		2.8	4	5	5	2.6	1.4

Considering the four performance metrics mentioned above, a cumulative evaluation may be more reasonable for a general view of the metric ranks given in Figure 6. Based on the comparison metrics used in this study, as seen in Figure 6, BBO is a better MhA than the others. However, it cannot be concluded that BBO is universally better than the others.

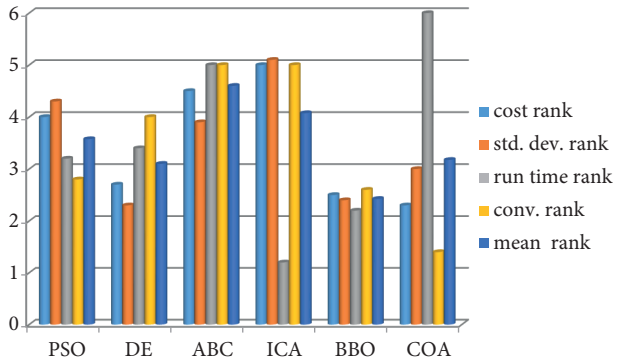

Figure 6. Comparison bar graphic based on the computed ranks.

The concept of performance is relative based on the needs of the user. For instance, if the running time is not important for the user, COA may be the best, but if it is important, ICA may be the best. Although MhAs are generally used in the design phase of systems, they can also be used for real-world applications with a hybrid structure like herein. Usage of MhAs for real-time applications needs fast computation. Due to this demand, fast processing platforms like FPGA are well suited for implementing hybrid systems. A good example of the real-time application of a hybrid system, as used here, is the work given in [15]. From this point of view, the comparison presented here also provides a general framework for users in terms of real-time applications. The good performance of BBO on the dynamic benchmark systems used here shows that BBO can be successfully applied to practical problems in engineering and science.

References

[1] Zheng YJ, Ling HF, Chen SY, Xue JY. A hybrid neuro-fuzzy network based on differential biogeography-based optimization for online population classification in earthquakes. IEEE T Fuzzy Syst 2015; 23: 1070-1083.

[2] Karaboğa D, Kaya E. Training ANFIS by using artificial bee colony (ABC) algorithm. Turk J Electr Eng & Comp Sci 2017; 25: 1669-1679.

[3] Kennedy J. Stereotyping: improving particle swarm performance with cluster analysis. In: 2000 Congress on Evolutionary Computation; 16–19 July 2000; La Jolla, CA, USA. New York, NY, USA: IEEE. pp. 1507-1512.

[4] Storn R, Price K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 1997; 11: 341-359.

[5] Abidin D. Genetic algorithm and differential evolution algorithm compared on a novel application domain. Trakya University Journal of Engineering Sciences 2016; 17: 11-22.

[6] Karaboğa D. An idea based on honey bee swarm for numerical optimization. Technical Report, Erciyes University, Kayseri Turkey, 2005.

[7] Akay B. Performance analysis of artificial bee colony algorithm on numerical optimization problems. PhD, Erciyes University, Kayseri Turkey, 2009 (thesis in Turkish with an abstract in English).

[8] Atashpaz-Gargari E, Lucas C. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: 2007 IEEE Congress on Evolutionary Computation; 25–28 September 2007; Singapore, Singapore. New York, NY, USA: IEEE. pp. 4661-4667.

[9] Simon D. Biogeography-Based Optimization. IEEE T Evolut Comput 2008; 12: 702-713.

[10] Rajabioun R. Cuckoo optimization algorithm. Appl Soft Comput 2011; 11: 5508-5518.

[11] Narendra KS, Parthasarathy K. Identification and control of dynamical systems using neural networks. IEEE T Neural Netwo 1990; 1: 4-27.

[12] Babuska R. Fuzzy Modeling for Control. Boston, MA, USA: Kluwer Academic Publisher, 1998.

[13] Oussar Y, Rivals I, Dreyfus L. Training wavelet networks for nonlinear dynamic input output modeling. Neurocomputing 1998; 20: 173-188.

[14] Sastry PS, Santharam G, Unnikrishnan KP. Memory neuron networks for identification and control of dynamical systems. IEEE T Neural Netwo 1994; 5: 306-319.

[15] Karakuzu C, Karakaya F, Çavuşlu MA. FPGA implementation of neuro-fuzzy system with improved PSO learning. Neural Networks 2016; 79: 128-140.

[16] Jang J-SR. ANFIS: Adaptive-network-based fuzzy inference system. IEEE T Syst Man Cyb 1993; 23: 665-685.

[17] Nobile MS, Pasi G, Cazzaniga P, Besozzi D, Colombo R, Mauri G. Proactive particles in swarm optimization: a self-tuning algorithm based on fuzzy logic. In: 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE); 2–5 Aug 2015; İstanbul, Turkey. New York, NY, USA: IEEE. pp. 1-8.