DOI: 10.15439/2016F206
ACSIS, Vol. 8. ISSN 2300-5963

Evaluation of selected fuzzy particle swarm optimization algorithms

Tomasz Krzeszowski, Krzysztof Wiktorowicz
Rzeszow University of Technology
Faculty of Electrical and Computer Engineering
al. Powstanców Warszawy 12, 35-959 Rzeszów
Email: {tkrzeszo, kwiktor}@prz.edu.pl

***Abstract*—This paper is devoted to an evaluation of selected fuzzy particle swarm optimization algorithms. Two non-fuzzy and four fuzzy algorithms are considered. The Takagi-Sugeno fuzzy system is utilized to change the parameters of these algorithms. A modified fuzzy particle swarm optimization method is proposed, in which each of the particles has its own inertia weight and coefficients of the cognitive and social components. The evaluation is based on the common nonlinear benchmark functions used for testing optimization methods. The ratings of the algorithms are assigned on the basis of the mean of the objective function and the relative success.**

I. Introduction

PARTICLE swarm optimization (PSO) is a stochastic optimization technique that was developed by Kennedy and Eberhart [1]. The PSO is mainly inspired by the social behavior of organisms that live and interact within large groups, for example, schools of fish, flocks of birds or swarms of bees. The usefulness of PSO in solving a wide range of optimization problems has been repeatedly confirmed. It has been applied to: the intelligent identification and control of a dynamic system [2]; solving an economic dispatch problem in power systems [3]; human motion tracking [4]; feature selection [5]; automatic incident detection [6]; fuzzy anomaly detection in networks [7]; the estimation of hurdles clearance parameters [8] and many more problems. Many variants of the PSO have been developed since it was introduced in 1995 [1]. The most common are algorithms with a constriction factor [9] and with a linear inertia weight [10]. Among the PSO modifications we can distinguish algorithms that utilize fuzzy systems [2], [3], [11]–[15]. For example, in papers [2], [11] a fuzzy system was used to dynamically modify the inertia weight. Another approach was presented in [3], where a fuzzy system is used to change the inertia weight and the coefficients of the cognitive and social components.

The goal of this study is to evaluate selected fuzzy PSO algorithms and to propose a modified fuzzy PSO algorithm. In our research, we use the Takagi-Sugeno system [16] instead of the Mamdani system [17] because it has shorter processing time. In this paper, we consider six different versions of PSO, including two non-fuzzy, and four fuzzy algorithms. The evaluation is based on nonlinear benchmarks in the form of Ackley, Griewank, Rastrigin and Rosenbrock functions. The calculations were conducted using Matlab software and the "PSO Research Toolbox" by Evers [18].

II. Particle swarm optimization

The particle swarm model consists of a group of particles that are randomly initialized in the d-dimensional search space. During an iterative process, particles explore this space effectively by exchanging information to find the optimal solution. Each i-th particle is described by its position $\mathbf{x}_i$, velocity $\mathbf{v}_i$, and best position $\mathbf{pbest}_i$. Moreover, the particles have access to the best global position $\mathbf{gbest}$ that has been found by any particle in the swarm.

In the basic PSO algorithm [1], the velocity and the position of each particle in k-th iteration are updated using the following equations:

$$\mathbf{v}_i^{k+1} = \mathbf{v}_i^k + c_1\mathbf{r}_1(\mathbf{pbest}_i^k - \mathbf{x}_i^k) + c_2\mathbf{r}_2(\mathbf{gbest}^k - \mathbf{x}_i^k) \quad (1)$$

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^{k+1} \quad (2)$$

where $\mathbf{r}_1$, $\mathbf{r}_2$ are vectors with uniformly distributed random numbers in the interval $[0, 1]$, and c_1, c_2 are positive constants equal to 2.

The velocities of particles are determined by three components. The first component is the inertia that models the particle's tendency to continue moving in the same direction. The second component is cognitive and attracts particles towards the best position previously found by the particle. The last component is a social component that moves particles towards the best position found earlier by any particle. Selection of the best global position and the best position for i-th particle is based on the objective function (denoted later by $f(\cdot)$).

A. PSO1: Clerc, Kennedy algorithm [9]

Many approaches have been developed to improve the performance of the basic PSO algorithm. One way is to use the constriction factor χ that was proposed by Clerc and Kennedy [9]. The application of this factor controls the velocity magnitude.

The velocity equation has the form:

$$\mathbf{v}_i^{k+1} = \chi[\mathbf{v}_i^k + c_1\mathbf{r}_1(\mathbf{pbest}_i^k - \mathbf{x}_i^k) + c_2\mathbf{r}_2(\mathbf{gbest}^k - \mathbf{x}_i^k)] \quad (3)$$

where χ is calculated as $\chi = \frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|}$ and $\varphi = c_1 + c_2$, $\varphi > 4$. In this paper, the following typical values are used: $c_1 = c_2 = 2.05$, $\varphi = 4.1$ and $\chi = 0.7298$.

978-83-60810-90-3/$25.00©2016, IEEE

B. PSO2: Eberhart, Shi algorithm [10]

Another way to improve the performance of PSO is to use the inertia weight ω. The inertia weight is significant for the performance of PSO, because it balances the global exploration and local exploitation abilities of the swarm. Exploration is facilitated when the inertia weight is high, but convergence is slower. On the other hand, when the inertia weight is low then convergence is faster, but it sometimes leads to local solutions. Hence, linearly decreasing inertia weight is proposed in [10].

The velocity equation has the form of

$$\mathbf{v}_i^{k+1} = \omega^k \mathbf{v}_i^k + c_1 \mathbf{r}_1 (\mathbf{pbest}_i^k - \mathbf{x}_i^k) + c_2 \mathbf{r}_2 (\mathbf{gbest}^k - \mathbf{x}_i^k) \quad (4)$$

The inertia weight ω is calculated by the formula

$$\omega^k = \omega_{max} - \frac{\omega_{max} - \omega_{min}}{iter_{max}} \cdot k \quad (5)$$

where ω_{max} is the initial weight, ω_{min} is the final weight and $iter_{max}$ is the maximum number of iterations. The limits for ω are set to $\omega_{max} = 0.9$ and $\omega_{min} = 0.4$.

III. Takagi-Sugeno system

Consider the Takagi-Sugeno (T-S) fuzzy system with two inputs y_1, y_2 and one output u. For the input y_1 we define m fuzzy sets A_i (Fig. 1), for which the vertices are placed in points p_i, where $i = 1, \ldots, m$. Similarly, for the input y_2, we define n fuzzy sets B_j with vertices in points q_j, where $j = 1, \ldots, n$. The coordinates p_i and q_j are written in the form of the vectors $\mathbf{p} = [p_i] = [p_1, \ldots, p_m]$ and $\mathbf{q} = [q_j] = [q_1, \ldots, q_n]$, respectively.

The output of the system is described by $m \cdot n$ fuzzy inference rules in the form of

$$R_{ij} : \text{ IF } y_1 \in A_i \text{ AND } y_2 \in B_j, \text{THEN } u = z_{ij} \quad (6)$$

where $z_{ij} \in \mathbb{R}$ is the consequent of the rule R_{ij}. The rules (6) are written in the following table:

$$\begin{array}{c|ccc} y_1 \backslash y_2 & B_1 & \ldots & B_n \\ \hline A_1 & z_{11} & \ldots & z_{1n} \\ \vdots & \vdots & \iddots & \vdots \\ A_m & z_{m1} & \ldots & z_{mn} \end{array} \quad (7)$$

The output u of the Takagi-Sugeno system is calculated as the weighted average of z_{ij} and determined by

$$u = TS(y_1, y_2) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij}(y_1, y_2) z_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij}(y_1, y_2)} \quad (8)$$

where $w_{ij}(y_1, y_2) = A_i(y_1) \cdot B_j(y_2)$ denotes the degree of fulfillment of the rule R_{ij}.

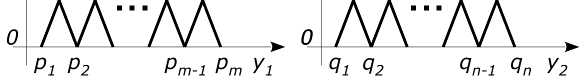

Fig. 1. Fuzzy sets for the inputs y_1 and y_2

IV. Fuzzy PSO

A. FPSO1: algorithm based on the work by Shi, Eberhart [11]

Better PSO performance can be obtained using the nonlinearly changing inertia weight that balances global and local search abilities. It is difficult to design a mathematical model to adapt the inertia weight dynamically. The solution to this problem may be obtained using a linguistic description of the search process. For example, we can use a fuzzy inference system for tuning the inertia weight [11].

In the FPSO1 algorithm, the inertia weight is described by the formula

$$\omega^{k+1} = \omega^k + \Delta\omega, \quad \Delta\omega = TS(nf^k, \omega^k) \quad (9)$$

where the T-S fuzzy system (8) is used to calculate the change of inertia weight $\Delta\omega$. The input nf^k is the normalized objective function value described by

$$nf^k = \frac{fg^k - f_{min}}{f_{max} - f_{min}} \quad (10)$$

where $fg^k = f(\mathbf{gbest}^k)$, f_{min} is the optimal solution (for the test functions considered in this paper, it is equal to 0), f_{max} is the worst solution (in our paper $f_{max} = f(\mathbf{gbest}^0)$). The fuzzy sets for the inputs nf^k and ω^k have vertices in points $\mathbf{p} = [0, 0.5, 1]$, $\mathbf{q} = [0.4, 0.7, 1]$, respectively, and the fuzzy rules have the form of

$$\begin{array}{c|ccc} nf^k \backslash \omega^k & B_1 & B_2 & B_3 \\ \hline A_1 & Z & N & N \\ A_2 & P & Z & N \\ A_3 & P & Z & N \end{array} \quad (11)$$

where $N = -0.1$, $Z = 0$ and $P = 0.1$.

B. FPSO2: algorithm based on the work by Alfi, Fateh [2]

The improvement of the FPSO1 algorithm was proposed by Alfi and Fateh [2]. In their method, the inertia weight is calculated for each particle according to its current state. This is justified because each particle in the swarm is in a different place in a complex environment and may have a different balance between global and local search abilities.

In the FPSO2 algorithm, the change of inertia weight is determined by the T-S system (8) as

$$\Delta\omega_i = TS(nf_i^k, \omega_i^k) \quad (12)$$

where

$$nf_i^k = \frac{fp_i^k - f_{min}}{fp_i^0 - f_{min}} \quad (13)$$

and $fp_i^k = f(\mathbf{pbest}_i^k)$. The vertices of fuzzy sets for nf_i^k and ω_i^k are chosen as $\mathbf{p} = [0, 0.5, 1]$, $\mathbf{q} = [0.4, 0.6, 0.8]$ respectively, and the fuzzy rules are

$$\begin{array}{c|ccc} nf_i^k \backslash \omega_i^k & B_1 & B_2 & B_3 \\ \hline A_1 & P & N & N \\ A_2 & P & Z & N \\ A_3 & P & Z & N \end{array} \quad (14)$$

where $N = -0.1$, $Z = 0$ and $P = 0.1$.

C. FPSO3: algorithm based on the work by Niknam [3]

In the FPSO3 algorithm, the fuzzy system proposed by Niknam [3] is used to change not only ω, but also the coefficients c_1 and c_2. These coefficients determine the influence of the personal best position $\mathbf{pbest}_i$ and the global best position $\mathbf{gbest}$ on the particle velocity. For example, if c_1 is larger than c_2, then the particle has the tendency to move to the personal best position, rather than to the global best position found by the swarm.

In the FPSO3 algorithm, three T-S systems are used to determine ω, c_1 and c_2:

$$\omega = TS(nf^k, nu^k) \tag{15}$$

$$c_1 = TS(nf^k, nu^k) \tag{16}$$

$$c_2 = TS(nf^k, nu^k) \tag{17}$$

The input nf^k is defined in (10) and nu^k is the normalized number of iterations without change of the best global position:

$$nu^k = \frac{u^k - u_{min}}{u_{max} - u_{min}} \tag{18}$$

where u^k is the number of iterations without change of the best global position, $u_{min} = 0$ and $u_{max} = iter_{max}$. The fuzzy sets are defined by the vectors $\mathbf{p} = [0.2, 0.4, 0.6, 0.8]$, $\mathbf{q} = [0.2, 0.4, 0.6, 0.8]$. The fuzzy rules for ω are defined as

$nf \setminus nu$	B_1	B_2	B_3	B_4
A_1	PS	PM	PB	PB
A_2	PM	PM	PB	PR
A_3	PB	PB	PB	PR
A_4	PB	PB	PR	PR

(19)

In table (19) we have $PS = 0.4$, $PM = 0.6$, $PB = 0.8$ and $PR = 1$. The fuzzy rules for c_1/c_2 are defined as

$nf \setminus nu$	B_1	B_2	B_3	B_4
A_1	PR/PR	PB/PB	PB/PM	PB/PM
A_2	PB/PB	PM/PM	PM/PS	PS/PS
A_3	PB/PM	PM/PM	PS/PS	PS/PS
A_4	PM/PM	PM/PS	PS/PS	PS/PS

(20)

In table (20) we have $PS = 1.2$, $PM = 1.4$, $PB = 1.6$ and $PR = 1.8$.

D. MFPSO: authors' proposition

The modified fuzzy PSO (MFPSO) algorithm proposed by the authors combines the previously described concepts developed by Alfi, Fateh [2] and Niknam [3]. In this algorithm, each of particles has its own coefficients ω, c_1 and c_2 changing according to the linguistic description represented by the fuzzy rules. In this way, each of the particles may be treated individually. For example, if a particle has found the new local best position $\mathbf{pbest}_i$, then the inertia weight ω should be decreased and the coefficients c_1 and c_2 should be increased. On the other hand, if $\mathbf{pbest}_i$ has not changed for a long time, then a better strategy would probably be to increase ω and decrease c_1 and c_2 to improve the ability of exploration.

In the MFPSO algorithm, the authors propose ω, and c_1, c_2 for each particle to be determined using three T-S systems:

$$\omega_i = TS(nf_i^k, nu_i^k) \tag{21}$$

$$(c_1)_i = TS(nf_i^k, nu_i^k) \tag{22}$$

$$(c_2)_i = TS(nf_i^k, nu_i^k) \tag{23}$$

where nf_i^k is defined in (13), nu_i^k has the form of

$$nu_i^k = \frac{u_i^k - u_{min}}{u_{max} - u_{min}} \tag{24}$$

and u_i^k is the number of iterations without change to the best personal position for the i-th particle. It should be noted that in equation (18), nu^k is calculated based on the global best position $\mathbf{gbest}$, whereas in equation (24) nu_i^k is calculated based on the personal best position $\mathbf{pbest}_i$. The vertices of fuzzy sets for nf_i^k and nu_i^k are defined as $\mathbf{p} = [0.2, 0.45, 0.65, 0.9]$, $\mathbf{q} = [0.2, 0.45, 0.65, 0.9]$.

The fuzzy rules for ω are the same as in (19). The fuzzy rules for c_1 and c_2 are given in tables (20). In (20) we have $PS = 1.4$, $PM = 1.7$, $PB = 1.9$ and $PR = 2.2$. For example, the rule R_{11} has the form

$$\begin{aligned} R_{11}: &\text{ IF } nf_i^k \in A_1 \text{ AND } nu_i^k \in B_1, \\ &\text{ THEN } \omega = PS \text{ AND } c_1 = PR \text{ AND } c_2 = PR \end{aligned} \tag{25}$$

and the rule R_{44} has the form

$$\begin{aligned} R_{44}: &\text{ IF } nf_i^k \in A_4 \text{ AND } nu_i^k \in B_4, \\ &\text{ THEN } \omega = PR \text{ AND } c_1 = PS \text{ AND } c_2 = PS \end{aligned} \tag{26}$$

Other rules can be interpreted similarly.

V. Results and discussion

In order to evaluate the algorithms, four common nonlinear benchmarks [11], [19] in the form of Ackley, Griewank, Rastrigin and Rosenbrock functions were used. For these functions, the asymmetric initialization method, similar to [11], was used. The velocities of particles were clamped to v_{max}, however, the positions of the particles were not limited. In Table I, the initialization ranges and v_{max} for the test functions are listed. In our experiments, two dimension sizes were chosen: $d = 10$ and $d = 30$. The number of iterations was set to 1000 and 2000 corresponding to the dimensions 10 and 30. The number of particles was equal to 30 and the number of trials was equal to 30 in all experiments. The parameters of algorithms were chosen on the basis of the papers [11] and [14]. The calculations were conducted using Matlab software and the "PSO Research Toolbox" by Evers [18]. The maximum average time of execution for one trial on a mobile workstation equipped with Intel(R) Core(TM) i7-2820QM did not exceeded 10 s.

The results for benchmark functions are presented in Table II. This table contains the basic statistics for the final value of the objective function and the iteration ($iter\_success$) in which the algorithm achieved the given value (th) of the

TABLE II
RESULTS FOR THE BENCHMARK FUNCTIONS

Algorithm	d	iter	fg mean ± std	fg min	fg max	iter_success mean ± std	iter_success min	iter_success max	success rate [%]
Ackley function: for $d = 10$, $th =$ 5e−05; for $d = 30$, $th = 5$									
PSO1	10	1000	2.223e−05 ± 1.218e−04	3.553e−15	6.669e−04	244 ± 23	212	300	96.7
	30	2000	8.218e+00 ± 7.791e+00	1.421e−14	1.980e+01	191 ± 51	141	343	56.7
PSO2	10	1000	6.685e−01 ± 3.662e+00	8.882e−14	2.006e+01	735 ± 21	697	777	96.7
	30	2000	6.909e−01 ± 3.784e+00	6.994e−07	2.073e+01	1073 ± 46	991	1191	96.7
FPSO1	10	1000	1.332e+00 ± 5.068e+00	3.553e−15	2.006e+01	216 ± 23	189	280	93.3
	30	2000	9.953e−01 ± 3.781e+00	1.421e−14	2.079e+01	472 ± 129	352	1004	96.7
FPSO2	10	1000	3.790e−15 ± 9.013e−16	3.553e−15	7.105e−15	257 ± 23	224	338	100
	30	2000	4.146e+00 ± 8.032e+00	2.807e−13	2.003e+01	257 ± 63	198	444	80.0
FPSO3	10	1000	9.000e−01 ± 3.662e+00	3.553e−15	2.006e+01	175 ± 152	118	883	80.0
	30	2000	8.351e+00 ± 7.650e+00	1.344e+00	1.998e+01	192 ± 59	126	362	66.7
MFPSO	10	1000	2.392e−14 ± 4.870e−14	3.553e−15	2.558e−13	366 ± 37	296	474	100
	30	2000	4.125e+00 ± 8.384e+00	1.028e−04	2.087e+01	440 ± 115	330	765	80.0
Griewank function: for $d = 10$, $th = 0.1$; for $d = 30$, $th = 0.05$									
PSO1	10	1000	7.258e−02 ± 3.584e−02	3.197e−02	2.115e−01	188 ± 80	109	483	86.7
	30	2000	2.701e−02 ± 4.005e−02	0.000e+00	1.858e−01	356 ± 31	299	435	83.3
PSO2	10	1000	1.050e−01 ± 5.726e−02	7.396e−03	2.172e−01	724 ± 129	541	974	46.7
	30	2000	1.303e−02 ± 1.775e−02	2.092e−11	9.064e−02	1524 ± 52	1463	1702	96.7
FPSO1	10	1000	8.642e−02 ± 3.961e−02	1.969e−02	1.796e−01	204 ± 153	76	569	70.0
	30	2000	1.375e−02 ± 1.688e−02	0.000e+00	5.888e−02	329 ± 38	292	476	93.3
FPSO2	10	1000	7.701e−02 ± 3.531e−02	3.201e−02	1.847e−01	234 ± 138	108	534	76.7
	30	2000	1.492e−02 ± 2.037e−02	0.000e+00	9.562e−02	446 ± 37	368	539	93.3
FPSO3	10	1000	9.796e−02 ± 5.344e−02	1.970e−02	2.511e−01	118 ± 75	56	348	56.7
	30	2000	3.036e+00 ± 2.411e+00	1.049e+00	1.128e+01	−	−	−	0
MFPSO	10	1000	9.623e−02 ± 5.589e−02	2.955e−02	2.488e−01	372 ± 201	145	823	60.0
	30	2000	1.956e−02 ± 2.617e−02	1.718e−06	1.298e−01	1184 ± 172	901	1507	93.3
Rastrigin function: for $d = 10$, $th = 5$; for $d = 30$, $th = 50$									
PSO1	10	1000	7.097e+00 ± 3.969e+00	1.990e+00	1.890e+01	235 ± 118	119	555	40.0
	30	2000	1.053e+02 ± 2.743e+01	4.676e+01	1.512e+02	330 ± 0	330	330	3.33
PSO2	10	1000	3.715e+00 ± 1.865e+00	0.000e+00	7.960e+00	717 ± 118	533	939	83.3
	30	2000	3.819e+01 ± 9.564e+00	2.389e+01	6.766e+01	1454 ± 128	1179	1655	93.3
FPSO1	10	1000	5.804e+00 ± 2.575e+00	2.985e+00	1.293e+01	229 ± 85	100	406	56.7
	30	2000	4.580e+01 ± 8.148e+00	3.084e+01	6.368e+01	486 ± 124	266	745	60.0
FPSO2	10	1000	4.743e+00 ± 3.394e+00	0.000e+00	1.791e+01	276 ± 153	100	690	66.7
	30	2000	4.852e+01 ± 1.391e+01	2.388e+01	8.457e+01	505 ± 174	256	869	56.7
FPSO3	10	1000	1.270e+01 ± 5.817e+00	9.950e−01	2.388e+01	117 ± 31	93	163	13.3
	30	2000	9.155e+01 ± 2.314e+01	5.330e+01	1.353e+02	−	−	−	0
MFPSO	10	1000	3.689e+00 ± 2.101e+00	4.832e−03	8.955e+00	447 ± 208	178	976	80.0
	30	2000	3.317e+01 ± 9.586e+00	1.435e+01	5.132e+01	960 ± 402	374	1814	96.7
Rosenbrock function: for $d = 10$, $th = 30$; for $d = 30$, $th = 100$									
PSO1	10	1000	2.155e+01 ± 3.702e+01	1.381e−02	1.261e+02	136 ± 81	64	391	80.0
	30	2000	3.793e+01 ± 5.813e+01	6.057e−02	2.642e+02	558 ± 419	255	1597	90.0
PSO2	10	1000	3.602e+01 ± 1.308e+02	6.977e−01	7.244e+02	613 ± 136	438	966	86.7
	30	2000	8.484e+01 ± 7.490e+01	5.490e+00	3.359e+02	1657 ± 177	1361	1996	66.7
FPSO1	10	1000	1.761e+01 ± 3.553e+01	2.459e−03	1.371e+02	239 ± 249	47	791	90.0
	30	2000	6.247e+01 ± 7.706e+01	3.930e−01	3.082e+02	499 ± 220	275	1252	76.7
FPSO2	10	1000	2.529e+01 ± 5.990e+01	7.227e−02	2.577e+02	171 ± 157	56	798	83.3
	30	2000	5.779e+01 ± 4.336e+01	1.420e+00	1.683e+02	688 ± 395	348	1992	83.3
FPSO3	10	1000	7.058e+01 ± 2.101e+02	2.104e+00	1.152e+03	133 ± 152	42	635	73.3
	30	2000	8.847e+04 ± 1.825e+05	2.877e+02	8.705e+05	−	−	−	0
MFPSO	10	1000	1.499e+01 ± 4.056e+01	1.937e−02	2.239e+02	276 ± 255	89	936	93.3
	30	2000	2.248e+02 ± 2.320e+02	1.188e+01	9.200e+02	1681 ± 186	1353	1936	36.7

TABLE I
PARAMETERS OF BENCHMARK FUNCTIONS

Function	Init. ranges	$\mathbf{v_{max}}$
Ackley	$(15, 30)^d$	30
Griewank	$(300, 600)^d$	600
Rastrigin	$(2.56, 5.12)^d$	5.12
Rosenbrock	$(15, 30)^d$	30

TABLE III
RATINGS OF THE PSO ALGORITHMS

	$\mathbf{d = 10}$		$\mathbf{d = 30}$		$\sum$	
Algorithm	**mfg**	**rs**	**mfg**	**rs**	**mfg**	**rs**
PSO1	16	18	11	17	27	35
PSO2	11	5	20	9	31	14
FPSO1	13	18	18	20	31	38
FPSO2	18	15	15	18	33	33
FPSO3	6	18	5	6	11	24
MFPSO	20	10	15	11	35	21

objective function. The ratings of the algorithms for all benchmark functions are summarized in Tab. III. The following performance measures were used to evaluate the algorithms:

- mean of the objective function (mfg),
- relative success defined as $rs = \frac{mean\ of\ iter\_success}{success\_rate}$.

For these measures, the sums of the ratings are shown in Tab. III. These ratings were assigned in such a way that the best algorithm has six points and the worst has one point. For $success\_rate = 0$ (the algorithm has not succeeded) the number of points is equal to zero.

For the dimension $d = 10$ and the measure mfg the highest rating has the algorithm MFPSO proposed by the authors, while for the measure rs the highest rating have the PSO1, FPSO1 and FPSO3. For the dimension $d = 30$ and the measure mfg the highest rating has the algorithm PSO2, while for the measure rs the highest rating has the FPSO1. Analyzing the sum of ratings for mfg it can be seen that the best algorithm is the MFPSO. For rs the MFPSO is the one before last. However, it should be emphasized that in the evaluation of optimization algorithms the most important criterion is the obtained objective function value.

VI. CONCLUSION

In this paper, the evaluation of selected fuzzy particle swarm optimization algorithms was presented. Two non-fuzzy and four fuzzy algorithms were considered. The main contributions of this paper are as follows:

- the application of the Takagi-Sugeno system that is more computationally efficient than the Mamdani system,
- a proposal for the use of the MFPSO algorithm, in which each of the particles has its own inertia weight and the coefficients of the cognitive and social components,
- the evaluation of selected fuzzy PSO algorithms using common benchmark functions.

Further work will focus on improving the proposed algorithm, building models to support the training process in sport [20], and the analysis of athletes' technique [8].

ACKNOWLEDGMENT

This work has been supported by the Polish Ministry of Science and Higher Education under grant No. U-722/DS/M.

REFERENCES

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. of IEEE Int. Conf. on Neural Networks*, vol. 4. IEEE Press, Piscataway, NJ, 1995, pp. 1942–1948.

[2] A. Alfi and M.-M. Fateh, "Intelligent identification and control using improved fuzzy particle swarm optimization," *Expert Systems with Applications*, vol. 38, no. 10, pp. 12 312–12 317, 2011.

[3] T. Niknam, "A new fuzzy adaptive hybrid particle swarm optimization algorithm for non-linear, non-smooth and non-convex economic dispatch problem," *Applied Energy*, vol. 87, no. 1, pp. 327–339, 2010.

[4] S. Saini, N. Zakaria, D. R. A. Rambli, and S. Sulaiman, "Markerless human motion tracking using hierarchical multi-swarm cooperative particle swarm optimization," *PLoS ONE*, vol. 10, no. 5, 2015.

[5] M. Adamczyk, "Parallel feature selection algorithm based on rough sets and particle swarm optimization," in *Computer Science and Information Systems (FedCSIS), 2014 Federated Conf. on*, Sept 2014, pp. 43–50.

[6] D. Srinivasan, W. H. Loo, and R. L. Cheu, "Traffic incident detection using particle swarm optimization," in *Swarm Intelligence Symposium. SIS '03. Proceedings of the IEEE*, April 2003, pp. 144–151.

[7] A. Karami and M. Guerrero-Zapata, "A fuzzy anomaly detection system based on hybrid PSO-Kmeans algorithm in content-centric networks," *Neurocomputing*, vol. 149, Part C, pp. 1253–1269, 2015.

[8] T. Krzeszowski, K. Przednowek, K. Wiktorowicz, and J. Iskra, "Estimation of hurdle clearance parameters using a monocular human motion tracking method," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 19, no. 12, pp. 1319–1329, 2016, PMID: 26838547.

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

[10] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 1, 2000, pp. 84–88.

[11] Y. Shi and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation*, vol. 1, 2001, pp. 101–106.

[12] A. M. Abdelbar, S. Abdelshahid, and D. C. Wunsch, "Fuzzy PSO: a generalization of particle swarm optimization," in *Proceedings. IEEE International Joint Conference on Neural Networks*, vol. 2, July 2005, pp. 1086–1091.

[13] H. Liu, A. Abraham, and W. Zhang, "A fuzzy adaptive turbulent particle swarm optimisation," *Int. J. Innov. Comput. Appl.*, vol. 1, no. 1, pp. 39–47, 2007.

[14] Y.-T. Juang, S.-L. Tung, and H.-C. Chiu, "Adaptive fuzzy particle swarm optimization for global optimization of multimodal functions," *Information Sciences*, vol. 181, no. 20, pp. 4539–4549, 2011, Special Issue on Interpretable Fuzzy Systems.

[15] J. J. D. Nesamalar, P. Venkatesh, and S. C. Raja, "Managing multi-line power congestion by using Hybrid Nelder-Mead - Fuzzy Adaptive Particle Swarm Optimization (HNM-FAPSO)," *Applied Soft Computing*, vol. 43, pp. 222–234, 2016.

[16] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *Systems, Man and Cybernetics, IEEE Transactions on*, no. 1, pp. 116–132, 1985.

[17] E. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975.

[18] G. Evers, "PSO Research Toolbox (Version 20110515), M.S. thesis code," 2016. [Online]. Available: http://www.georgeevers.org/pso_research_toolbox.htm

[19] J. J. Liang, P. N. Suganthan, and K. Deb, "Novel composition test functions for numerical global optimization," in *Proceedings. IEEE Swarm Intelligence Symposium. SIS 2005*, June 2005, pp. 68–75.

[20] K. Wiktorowicz, K. Przednowek, L. Lassota, and T. Krzeszowski, "Predictive modeling in race walking," *Computational Intelligence and Neuroscience*, vol. 2015, p. 9, 2015, Article ID 735060.