sciendo

JAISCR, 2020, Vol. 10, No. 2, pp. 95 – 111
10.2478/jaiscr-2020-0007

A NEW AUTO ADAPTIVE FUZZY HYBRID PARTICLE SWARM OPTIMIZATION AND GENETIC ALGORITHM

Piotr Dziwiński[1,*], Łukasz Bartczuk[1], Józef Paszkowski[2,3]

[1] *Department of Computational Intelligence, Czestochowa University of Technology, al. Armii Krajowej 36, 42-200 Częstochowa, Poland*

[2]*Information Technology Institute, University of Social Sciences, 90-113 Łódź*

[3]*Clark University Worcester, MA 01610, USA*

[*]*E-mail: piotr.dziwinski@pcz.pl*

Submitted: 29th September 2019; Accepted: 21st February 2020

Abstract

The social learning mechanism used in the Particle Swarm Optimization algorithm allows this method to converge quickly. However, it can lead to catching the swarm in the local optimum. The solution to this issue may be the use of genetic operators whose random nature allows them to leave this point. The degree of use of these operators can be controlled using a neuro-fuzzy system. Previous studies have shown that the form of fuzzy rules should be adapted to the fitness landscape of the problem. This may suggest that in the case of complex optimization problems, the use of different systems at different stages of the algorithm will allow to achieve better results. In this paper, we introduce an auto adaptation mechanism that allows to change the form of fuzzy rules when solving the optimization problem. The proposed mechanism has been tested on benchmark functions widely adapted in the literature. The results verify the effectiveness and efficiency of this solution.

Keywords: hybrid methods, Particle Swarm Optimization, Genetic Algorithm, fuzzy systems, multimodal functions

1 Introduction

The solution of almost any engineering or scientific problem requires optimizing the parameters to find the solution that minimizes costs and maximizes efficiency. The optimization task can be solved using many different methods. Simple, low dimensional optimization problems can be solved analytically, which allows to obtain an accurate solution. However, many real-world optimization problems are multidimensional, multimodal or noisy so that they are too complex for these methods to be effective. Such a task can be solved with the help of heuristic methods, which, allow to get an approximate (however, not necessarily the best) solution. These methods can be divided into two groups:

- deterministic optimization methods,
- randomized optimization methods.

Initially, most deterministic methods often choose from the solution space one random solution, which is then modified in a way that brings them closer to the optimal solution. Their advantage is that with the same parameters and starting

from the same starting point, they allow to get the same results every time.

In turn, randomized methods, to accelerate the process of finding the optimal solution, introduce different types of random modification of solutions. The most popular solutions of this type include evolutionary and swarm intelligence based algorithms, whose representatives are, respectively, genetic algorithms and PSO algorithm.

The PSO algorithm seems to be particularly popular. In this method, when a particle is modified, its current position, its best historical position and the best position found so far throughout the swarm are taken into account. This mechanism allows for fast convergence. However, it can also cause the global solution to get trapped in the local minimum due to too fast transition from the exploration phase (searching for regions that can potentially contain an optimal solution) to exploitation (searching for a solution in a limited space).

It can, therefore, be concluded that to improve search efficiency, an appropriate compromise should be found between exploration and exploitation. For this reason, in recent years there has been an increase in interest in hybrid methods combining the PSO mechanism with other algorithms, e.g. Local Searching Strategy [3], Harmony Search [21], Nelder-Mead simplex search [10, 11], Simulated Annealing [36], supervised learning and control [6] or population-based methods like Bat Algorithm [24], Cuckoo Search Algorithm [5] and Gray Wolf Optimization [30].

However, it seems that most often the PSO algorithm is combined with the genetic algorithm, because of the random nature of the crossing and mutation operations. That random nature allows this method to leave the local minimum. This combination can be implemented in several different ways:

1. *One algorithm is used to initialize the other*. In paper [33] Tang et al. proposed to run the PSO algorithm first and use the solutions found as an initial population for the GA algorithm. Robinson et al. [26] considered two ways to combine the GA and PSO method. In the first one, the GA population was used to start the PSO, and in the second one, the PSO swarm was used as the initial population of GA.

2. *Different algorithms operate simultaneously in an independent manner*. Shi et al. [28] proposed an integration scheme, where the PSO and GA algorithms work in parallel and after reaching a certain number of iterations, exchange individuals found by both methods. Valdez et al [34] hybridized GA and PSO by using a fuzzy logic system for decision-making. In this method, the authors used three different fuzzy systems making decisions based on the current value of the error and its derivative. The first system determines which PSO algorithm or GA algorithm should be used in further computations. The other two are used to choose parameters of PSO or GA algorithms.

3. *The main loop of the algorithm executes different methods sequentially*. In paper [1], researchers suggested an approach where PSO and GA algorithms are run alternately on some number of iterations. The GA algorithm is also used to co-evolve the population of infeasible individuals until they become feasible.

4. *In one iteration, solutions are modified by operators from different methods*. This is the most popular schema of hybridization. In paper [14], Gong et al. demonstrated the scheme in which the main loop of the algorithm consists of two cascading GA and PSO layers. Kuo and Han [20] suggested using the mutation operator if the PSO is stagnating (the personal and global solutions do not change). Other approaches from this group can be found in [16, 12, 17, 25].

In our previous article [8], we proposed a hybrid method (FSHPSO-E). This method combines PSO and genetic algorithms, while the impact of the latter on the process of searching for the optimal solution is determined by the influence factor. Its value is determined using a fuzzy system and changes during the operation of the algorithm.

It should be noted that various problems (or at least their groups, e.g. unimodal and multimodal problems) may require different ways of modifying the influence factor. Moreover, searching for their solution strongly depends on the selected parameters of PSO and genetic algorithms. In practice, however, it is difficult to determine a priori the nature of the optimization problem. From this reason,

we believe that the introduction of a mechanism that will allow the automatic selection of appropriate parameters of the fuzzy system and the PSO and GA algorithms can improve the efficiency of the search process for the optimal solution. This mechanism is the main contribution of this work.

The proposed algorithm is explained in detail in Section 3, after the brief introduction to the PSO method, the Generic Algorithms and the hybrid FSHPSO-E method in Section 2. We validate the proposed method on the set of well-known unimodal and multimodal benchmark functions. The results of the simulation, which we show in Section 4, indicate the promising performance.

2 The Hybrid Algorithm Combining PSO and GA Controlled by Fuzzy Logic

The PSO and GA algorithms are two very popular methods to solve nonlinear optimization problems in the form

$$\begin{array}{ll} \text{Minimize} & f(\mathbf{x}) \\ \text{subjected to} & \begin{cases} g_i(\mathbf{x}) \leq 0 & i=1,\cdots,P \\ h_j(\mathbf{x}) = 0 & j=1,\cdots,Q \\ x_d \in [\underline{x^d}, \overline{x^d}] & d=1,\ldots,D \end{cases} \end{array}, \tag{1}$$

where: f is an optimized objective function, $\mathbf{x} = [x^1,\ldots,x^D]^T$ is a D dimensional solution of the problem, $\underline{x^d}$ and $\overline{x^d}$ are the minimum and maximum permissible values for the d-th variable respectively; $h_j(\mathbf{x})$ and $g_i(\mathbf{x})$ define equality and inequality constraints.

2.1 The Particle Swarm Optimization Algorithm

The PSO algorithm has been proposed by Eberhart and Russel in 1995 [9]. It is an interative method that processed the entire collection (the swarm) of potential solutions of the problem stated by (1). Each potential solution is represented as a particle $a_i(t) = (\mathbf{x}_i(t), \mathbf{v}_i(t), \mathbf{p}_i(t))$ where $\mathbf{x}_i(t)$ is a specific position in solution space $\mathbf{x}_i(t) = [x_i^1(t),\ldots,x_i^D(t)]$, $\mathbf{v}_i(t)$ is a non-zero velocity $\mathbf{v}_i(t) = [v_i^1(t),\ldots,v_i^D(t)]$ and $\mathbf{p}_i(t)$ is a historical best personal position $\mathbf{p}_i(t) = [p_i^1(t),\ldots,p_i^D(t)]$.

The position and velocity of particles are modified in each iteration of the algorithm according to the following equations

$$\begin{aligned} v_i^d(t+1) &= w \cdot v_i^d(t) \\ &\quad + \psi_1 \cdot r_1 \cdot (p_i^d(t) - x_i^d(t)) \\ &\quad + \psi_2 \cdot r_2 \cdot (g^d(t) - x_i^d(t)), \end{aligned} \tag{2}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t), \tag{3}$$

where w is an inertia weight $w \in (0,1]$, ψ_1 and ψ_2 are constants defined in advance and r_1 and r_2 are uniform random numbers within the range $(0,1]$. When determining the new velocity of the particle, its best historical position (personal impact) and globally best solution $\mathbf{g}(t) = [g^1(t),\ldots,g^D(t)]$ of whole swarm or some topological neighbourhood of p_i (social impact) are taken into account. The ψ_1 and ψ_2 values allow to control the influence of the personal and social parts of the equation (2).

As explained in [9], this method of particle modification was inspired by the social behaviour of animals like fish schooling or bird flocking.

2.2 The Genetic Algorithm

Another well known optimization method is the genetic algorithm [13, 18, 27]. This method also processes the whole collection of potential solutions (the population), however, is inspired by natural evolution. At the beginning, the algorithm creates the initial population of μ random potential solutions (the individuals). Then, in each iteration, the reproduction operator randomly creates a temporary population with λ individuals. Individuals from this population are modified by the crossover and mutation operators (ensuring exploitation and exploration of the solution space). Afterwards, the μ best individuals (in the sense of objective function) are passed to the next iteration. Algorithm 2 presents the general scheme of the genetic algorithm. In this method, each potential solution of the problem defined by (1) is presented as a chromosome $\mathbf{C}_j = [c_j^1,\ldots,c_j^D], c^d \in [\underline{c^d}, \overline{c^d}], d = 1,\ldots,D$, $j = 1,\ldots,|P|$ where $|P| = \mu$ in case of parental population Pop and $|P| = \lambda$ for the temporary population Temp. There are many different methods of reproducing individuals, as well as ways to implement crossover and mutation operators. In this paper, we use tournament selection [4, 31], and the

crossover and mutation operators defined by the equations

$$\mathbf{C}_{j1} = \begin{cases} \mathbf{C}_{j1}^k & \text{if } k \in \{k_1, \ldots, k_{n_c}\} \\ \mathbf{C}_{j2}^k & otherwise \end{cases}, \qquad \mathbf{C}_{j2} = \begin{cases} \mathbf{C}_{j2}^k & \text{if } k \in \{k_1, \ldots, k_{n_c}\} \\ \mathbf{C}_{j1}^k & otherwise \end{cases}, \tag{4}$$

$$\mathbf{C}_j = \begin{cases} \phi \cdot (\overline{c^k} - \underline{c^k}) + \underline{c^k} & \text{if } k \in \{k_1, \ldots, k_{n_m}\} \\ \mathbf{C}_j^k \end{cases}, \tag{5}$$

where j_1, j_2, j are randomly chosen from the set $\{1, \ldots, \lambda\}$, $k = 1, \ldots, D$ points to the index of gene in chromosome, $\{k_1, \ldots, k_n\}$, is a random set of distinct gene indexes that will be exchanged between parents by the crossover operator or modified by the mutation operator, $n \in \{n_c, n_m\}$, n_c, n_m are the random numbers of genes that will be exchanged or modified respectively and ϕ is a random number from the uniform distribution $U(0,1)$.

2.3 The FSHPSO-E Algorithm

Both of the presented algorithms allow to achieve good results and have been successfully used to solve problems in many different areas (see e.g. [2, 19, 32, 38]). However, their mechanisms can cause a reduction in population diversity, which can lead to premature convergence to the local, suboptimal extreme.

In paper [7], we presented a hybrid algorithm (HPSO-E) that integrates the PSO method with the genetic algorithm. In this method in one iteration of the main loop of the algorithm, the PSO operators are applied N times, modifying all particles from the swarm $\mathbf{S}(t)$; where N means the size of the PSO swarm. The genetic operators can be executed only the $\lfloor p_e \cdot N \rfloor$ times where $p_e \in [0,1]$ means the influence factor that determines the impact of the genetic algorithm on the search process.

In the article [8], we extended this idea and assumed that the value of the influence factor should not be constant, but it changes depending on the current state of the PSO algorithm. We also proposed using the fuzzy system to control p_e and, as a result, the impact of genetic operators on the process of searching the global optima. Figure 1 presents the general idea of this method, which we called FSHPSO-E.

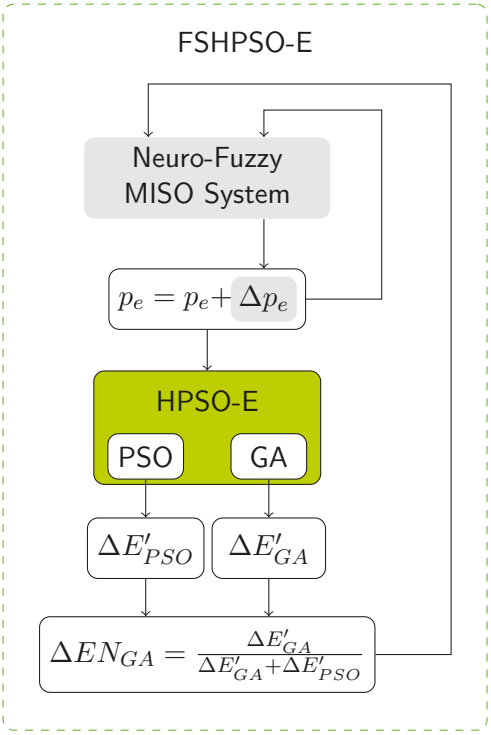

Figure 1. The general idea of the FSHPSO-E algorithm

To modify the p_e value, after every w_m iterations we determine the normalized efficiency ΔEN_{GA} of genetic to PSO operators, which can be represented by the formula

$$\Delta EN_{GA} = \frac{\Delta E'_{GA}}{\Delta E'_{GA} + \Delta E'_{PSO}}, \tag{6}$$

where $\Delta E'_{GA}$ and $\Delta E'_{PSO}$ determine the efficiency of GA and PSO defined as the average improvement (decrease) of the fitness function during the last w_o iterations and computed according to equations

$$\Delta E'_{GA} = \frac{\sum_{t'=t-w_o}^{t} E_{GA}(t')}{\sum_{t'=t-w_o}^{t} |\mathbf{CH}(t')|}, \tag{7}$$

$$\Delta E'_{PSO} = \frac{\sum_{t'=t-w_o}^{t} E_{PSO}(t')}{w_o \cdot N}. \tag{8}$$

$|\mathbf{CH}(t')|$ is the size of temporary population, created by genetic operators, in the iteration t', $E_{GA}(t')$ and $E_{PSO}(t')$ mean the effectiveness of GA and PSO

in the iteration t' and are defined as a total improvement (decrease) of the fitness of the best solution

$$E_{GA}(t') = \sum_{j=1}^{|\mathbf{CH}(t)|} \begin{cases} f(\mathbf{g}(t')) - f(\mathbf{o}_j)) \\ \quad \text{if } f(\mathbf{g}(t')) > f(\mathbf{o}_j), \\ 0 \quad \text{otherwise} \end{cases} \tag{9}$$

$$E_{PSO}(t') = \sum_{i=1}^{N} \begin{cases} f(\mathbf{g}(t')) - f(\mathbf{x}_i(t')) \\ \quad \text{if } f(\mathbf{g}(t')) > f(\mathbf{x}_i(t')), \\ 0 \quad \text{otherwise} \end{cases} \tag{10}$$

where $\mathbf{o}_j$ and $\mathbf{x}_i(t')$ mean the new elements obtained as the result of genetic operators and PSO modification, respectively. It should be noted that these values are affected only by solutions that improve the global best solution in the t' iteration.

The high value of ΔEN_{GA} indicates that genetic operators can create better solutions (in the sense of fitness function) than the PSO. If such a situation occurs in subsequent iterations, it may mean that the PSO algorithm stagnates or is stuck in the local minimum.

Based on information about the current value of the p_e and normalized efficiency of genetic algorithm ΔEN_{GA} the fuzzy system determines the value Δp_e which increases or decreases the influence the genetic algorithm on the process of searching for the optimal solution according to the following formula

$$p_e = p_e + \Delta p_e = p_e + FS(\Delta EN_{GA}, p_e). \tag{11}$$

In paper [8], we use the well-known neuro-fuzzy system of Mamdani type. For each input, we have defined three fuzzy sets which denote small, medium and large value with membership function of class L [27], triangular [27] and γ [27], respectively. For output, we have defined five fuzzy sets described by the singleton membership function [27]. Figure 2 shows the graphical illustration of initial fuzzy sets defined for each input and output. Based on these fuzzy sets, we defined nine fuzzy rules presented in Table 1.

Table 1. The set of rules of Mamdani neuro-fuzzy system used in the proposed method.

		ΔEN_{GA}		
		small	medium	large
p_e	small	decrease a bit(B2)	increase a bit (B4)	increase (B5)
	medium	decrease a bit(B2)	do nothing (B3)	increase a bit (B4)
	large	decrease (B1)	decrease a bit (B2)	increase a bit (B4)

The applied defuzzification method is the centre of averages which allows to define the output of the system as

$$\Delta p_e = \frac{\sum_{k=1}^{|R|} y^k \cdot \mu_{B^k}(y^k)}{\sum_{k=1}^{|R|} \mu_{B^k}(y^k)}, \tag{12}$$

where $|R|$ is the number of rules of the fuzzy system and $\mu_{B^k}(y^k)$ and y^k are a membership function and the output result of k-th rule respectively.

The singleton membership functions simplify the structure of the used system because the value Δp_e is independent of the type of the membership function of the output fuzzy sets. Their use also makes that the Mamdani type fuzzy system [27] is equivalent to a zero-order Takagi-Sugeno type fuzzy system [15].

It should be noted that to determine the effectiveness of algorithms reliably, the results from the observation window - the last w_o iterations of the main loop of the proposed method - are taken into account for the calculation of $\Delta E'_{GA}$ and $\Delta E'_{PSO}$. Also, to observe how the change in the p_e parameter affected the process of searching for the optimal solution, its value is modified not more often than every w_m iterations (modification window).

The significant element of the HPSO-E and FSHPSO-E methods is the merging strategy which is used to merge the temporary population with the PSO swarm $\mathbf{S}(t)$. It allows that particle $\mathbf{o}_i \in \mathbf{CH}(t)$ to replace its parent $\mathbf{x}_i$ from PSO swarm $\mathbf{S}(t)$ if and only if it is better (in the sense of fitness function) than $\mathbf{p}_i(t)$

$$\mathbf{x}_i(t) = \begin{cases} \mathbf{o}_i & \text{if} f(\mathbf{o}_i) < f(\mathbf{p}_i(t)) \\ \mathbf{x}_i(t) & otherwise, \end{cases} . \tag{13}$$

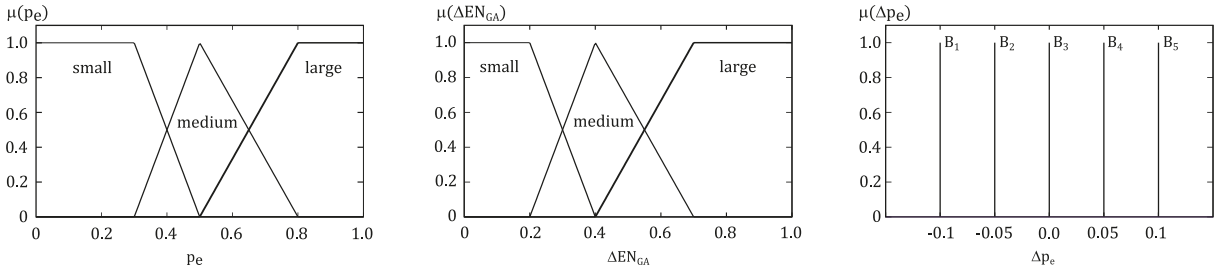

Figure 2. The graphical illustration of fuzzy sets of th initial system used in FSHPSO-E

The purpose of this strategy is to minimize the influence of genetic operators on swarm's dynamic. The pseudocode of the FSHPSO-E method is presented as algorithms 1 and 2.

Algorithm 1 FSHPSO-E algorithm

/* Initialization */
$t \leftarrow 0$
for $i \leftarrow 1$ to N **do**
  Randomly initialize $\mathbf{x}_i(t)$ and $\mathbf{v}_i(t)$
  $\mathbf{p}_i(t) \leftarrow \mathbf{x}_i(t)$
  Evaluate $f(\mathbf{x}_i(t))$
Set $\mathbf{g}(t)$

while Term. cond. has not been met **do**

  for $i \leftarrow 1, N$ **do** ▷ Particle update
    Modify $\mathbf{v}_i(t+1)$
    Modify $\mathbf{x}_i(t+1)$
    Evaluate $f(\mathbf{x}_i(t))$
    Update $\mathbf{p}_i(t)$

  $\mathbf{CH}(t) \leftarrow$CreateTemp$(\mathbf{S}(t), p_e, p_m, p_c, T)$

  Combine $\mathbf{S}(t)$ with $\mathbf{CH}(t)$
  Update $\mathbf{p}_i(t)$ and $\mathbf{g}(t)$

  Compute $E_{GA}(t)$ and $E_{PSO}(t)$
  if $t \mod w_m = 0$ **then**
    Compute ΔE_{GA}
    $p_e \leftarrow p_e + \Delta p_e = p_e + FS(\Delta E_{GA}, p_e)$
  $t \leftarrow t + 1$
Select the best global solution $\mathbf{g}(t_{max})$

3 FSHSPO-E algorithm with autoadaptive mechanism

Using the FSHPSO-E algorithm to solve the optimization problem, we will achieve the best results by appropriately choosing the form of fuzzy system rules and the parameters of the PSO and GA algorithms. The combination of a fuzzy system and parameters of the PSO and GA algorithm we call a search strategy and define as

$$\mathcal{ST} = \left\langle FS, [\underbrace{\omega, \psi_1, \psi_2, n_c, n_m, p_c, p_m}_{\text{parameters of PSO and GA}}] \right\rangle.$$

The choice of the right search strategy depends on the specific optimization problem and is not a trivial task. Also, it should be noted that some problems can be solved more effectively by changing the strategy used while the algorithm is running.

Algorithm 2 Create Temporary Population

function CreateTemp$(\mathbf{S}(t), p_e, p_m, p_c, T)$
  Set $\mathbf{CH} = \emptyset$
  for $m \leftarrow 1, \lfloor p_e \cdot N \rfloor$ **do**
    if $(p_m > r(0,1))$ **then** ▷ Mutation
      $\mathbf{p}_i(t) \leftarrow$ Tournament$(\mathbf{S}(t), T)$
      $\mathbf{o}_i \leftarrow Mutate(\mathbf{p}_i(t))$
      Evaluate $f(\mathbf{o}_i)$
      Insert $\mathbf{o}_i$ into $\mathbf{CH}$

    if $(p_c > r(0,1))$ **then** ▷ Crossover
      $\mathbf{p}_{i1}(t) \leftarrow$ Tournament$(\mathbf{S}(t), T)$
      $\mathbf{p}_{i2}(t) \leftarrow$ Tournament$(\mathbf{S}(t), T)$
      $(\mathbf{o}_{i1}, \mathbf{o}_{i2}) = Crossover(\mathbf{p}_{i1}, \mathbf{p}_{i2})$
      Evaluate $f(\mathbf{o}_{i1})$, $f(\mathbf{o}_{i2})$
      Insert $\mathbf{o}_{i1}$ and $\mathbf{o}_{i2}$ into $\mathbf{CH}$
  return CH

In this Section, we introduce a simple mechanism for automatic selection of a strategy from a set of available strategies (we will limit ourselves to two strategies only, although extending to more number can be implemented easily).

Let's assume that we have two search strategies $\mathcal{ST}_1$ and $\mathcal{ST}_2$. In each iteration of the algorithm, the potential solutions can be modified by a randomly selected strategy according to the specified probability $P(\mathcal{ST}_1) = \eta$ and $P(\mathcal{ST}_2) = 1 - \eta$. If we have more strategies, we can apply some more advanced selection procedure like the roulette wheel method [13]. At the beginning $\eta = 0.5$, and then it is modified during the progress of the algorithm based on the effectiveness of strategies determined by the following formulas

$$\eta = \frac{\Delta E'_{\mathcal{ST}_1}}{\sum_{k=1}^{2} \Delta E'_{\mathcal{ST}_k}}, \tag{14}$$

where $\Delta E'_{\mathcal{ST}_k}$ determine the efficiency of k-th strategy; $k = 1, 2$, computed according to equations

$$\Delta E'_{\mathcal{ST}_k} = \frac{\sum_{t'=t-w_n}^{t} E_{\mathcal{ST}_k}(t')}{T_{\mathcal{ST}_k}}, \tag{15}$$

$E_{\mathcal{ST}_k}(t')$ is the total improvement of the fitness of the best solution computed with formula

$$E_{\mathcal{ST}_k}(t') = \sum_{l=1}^{|\mathbf{O}(t')|} \begin{cases} f(\mathbf{g}(t')) - f(\mathbf{o}_l)) \\ \quad \text{if } f(\mathbf{g}(t')) > f(\mathbf{o}_l) \\ \quad \text{and } \mathcal{ST}_k = \mathcal{ST}_{t'} \\ 0 \quad \text{otherwise} \end{cases}. \tag{16}$$

$T_{\mathcal{ST}_k}$ is the number of evaluations of the fitness function during the application of the strategy $\mathcal{ST}_k$ defined as

$$T_{\mathcal{ST}_k} = \sum_{t'=t-w_n}^{t} \begin{cases} |\mathbf{O}(t')| & \text{if } \mathcal{ST}_k = \mathcal{ST}_{t'} \\ 0 & \text{otherwise} \end{cases}, \tag{17}$$

and $\mathbf{O}(t') = \mathbf{S}(t') \cup \mathbf{CH}(\mathbf{t}')$ is the set of solutions created in the t-th iteration.

Just like when evaluating the effectiveness of the PSO and GA algorithms in FSHPSO-E method, to obtain reliable statistics determined using formulas (14) - (16), they are calculated in w_n iterations.

The pseudocode of the proposed FSHPSO-E-AA method is presented as Algorithm 3.

Algorithm 3 FSHPSO-E-AA algorithm

```
   /* Initialization */
 1: t ← 0
 2: for i ← 1 to N do
 3:     Randomly initialize x_i(t) and v_i(t)
 4:     p_i(t) ← x_i(t)
 5:     Evaluate f(x_i(t))
 6: Set g(t)

 7: while Term. cond. has not been met do

 8:     if U(0,1) < η then
 9:         ST_t ← ST_1
10:     else
11:         ST_t ← ST_2

12:     for i ← 1, N do                    ▷ Particle update
13:         Modify v_i(t+1)
14:         Modify x_i(t+1)
15:         Evaluate f(x_i(t))
16:         Update p_i(t)

17:     CH(t) ← CreateTemp(S(t),
                  p_e, p_m^{ST_t}, p_c^{ST_t}, T)

18:     Combine S(t) with CH(t)
19:     Update p_i(t) and g(t)

20:     Compute E_GA(t) and E_PSO(t)
21:     if t mod w_m = 0 then
22:         Compute ΔE_GA
23:         p_e ← p_e + Δp_e = p_e + FS^{ST_t}(ΔE_GA, p_e)

24:     Compute E_{ST_t}(t) and E_{ST_t}(t)
25:     if t mod w_n = 0 then
26:         Compute ΔE'_{ST_k}, k = 1, 2
27:         Compute η
28:     t ← t + 1
29: Select the best global solution g(t_max)
```

4 Simulation results

To estimate the performance of the proposed auto-adaptive mechanism we have used the set of fifteen well-known benchmarks functions taken from [37]. This set includes continuous unimodal functions (F_1 - F_4), a step function (F_6), a noisy function (F_7), multimodal functions (F_5, F_8-F_{13}), shifted multimodal functions (F_{14}-F_{15}). After the

work [37], the Rosenbrock function (F_5) is included in the group of unimodal functions (in order to maintain the same numbering). However, it should be remembered that for $D > 3$ this function is a multimodal function. The equations and search ranges of used functions as well as the positions of global optima are presented in the Appendix.

4.1 Selection of the best combination of two strategies

In the first experiment, we want to determine which combination of two strategies can achieve the best results. For this purpose, for functions F_1-F_{13}, the GPSO algorithm sets the parameters of fuzzy system and parameters of the GA and PSO algorithms according to procedure described in [8].

Then we checked the performance of all possible pairs of different strategies. To obtain reliable results, the simulation for each benchmark function has been repeated 30 times. It should be noted that because all benchmark functions can be divided into unimodal and multimodal, the number of simulations can be reduced by combining only strategies prepared for different groups.

Figures 3-4 show the obtained simulation results. The heat maps presented in 3 and 4 are based on the p-value obtained in the hypothesis test. If the p-value is less than the significance level $\alpha = 0.05$, the difference of results is statistically significant. Figure 3 depicts how many times a given connection was statistically better than others ($p > 0.95$), and Figure 4 how many times this connection was significantly worse.

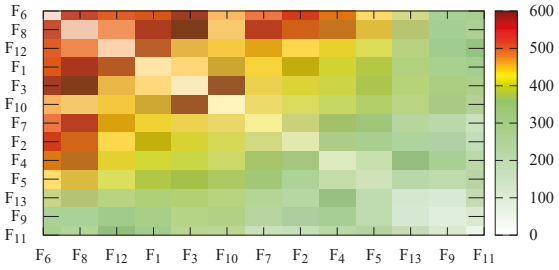

Figure 3. Comparison of different connections - significantly best (bigger value is better)

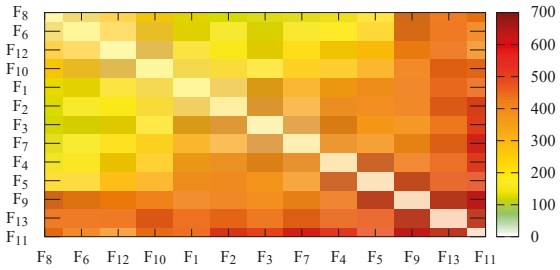

Figure 4. Comparison of different connections - significantly worse (lower value is better)

Table 3. Parameters of the PSO and GA algorithms contained in selected strategies

	ω	ϕ_1	ϕ_2	c_p	n_c	m_p	n_m	w_m	w_o
F_3	0.55	1.75	1.7	0.14	7	0.24	14	274	262
F_8	0.71	1.74	1.67	0.32	16	0.45	2	450	66

Table 4. The results obtained by the proposed FSHPSO-E-AA with F_3 and F_8 strategies and FSHPSO-E algorithm using fuzzy systems F_3 and F_8

Function		F_3	F_8	FSHPSO-E-AA
F_1	Mean	2.04e-002	2.47e-113	**3.13e-166**
	Best	3.31e-027	3.28e-130	**1.98e-323**
F_2	Mean	7.41e-002	**3.13e-062**	4.47e-040
	Best	1.18e-004	1.76e-069	**4.20e-084**
F_3	Mean	**2.19e-021**	6.49e-009	1.17e-019
	Best	1.42e-029	1.89e-011	**4.09e-035**
F_4	Mean	1.52e-003	1.84e-005	**1.18e-007**
	Best	**5.33e-010**	4.14e-007	5.86e-010
F_5	Mean	3.00e+001	1.48e+001	**1.22e+000**
	Best	1.64e+001	1.59e-001	**2.45e-004**
F_6	Mean	1.87e+000	**0.00e+000**	**0.00e+000**
	Best	**0.00e+000**	**0.00e+000**	**0.00e+000**
F_7	Mean	9.47e-003	**3.09e-003**	3.56e-003
	Best	2.09e-003	**1.35e-003**	1.61e-003
F_8	Mean	4.53e+000	**1.34e-002**	**1.34e-002**
	Best	**1.34e-002**	**1.34e-002**	**1.34e-002**
F_9	Mean	4.85e+000	1.59e+000	**1.01e-015**
	Best	1.49e-006	**0.00e+000**	**0.00e+000**
F_{10}	Mean	2.07e+000	2.72e-001	**8.63e-002**
	Best	4.16e-005	**7.55e-015**	8.63e-002
F_{11}	Mean	4.61e-002	3.84e-002	**3.75e-002**
	Best	**0.00e+000**	**0.00e+000**	**0.00e+000**
F_{12}	Mean	1.93e-001	3.57e-002	**1.63e-032**
	Best	4.33e-011	**1.62e-032**	**1.62e-032**
F_{13}	Mean	1.14e-001	1.15e+004	**1.85e-003**
	Best	3.46e-020	**1.35e-032**	**1.35e-032**

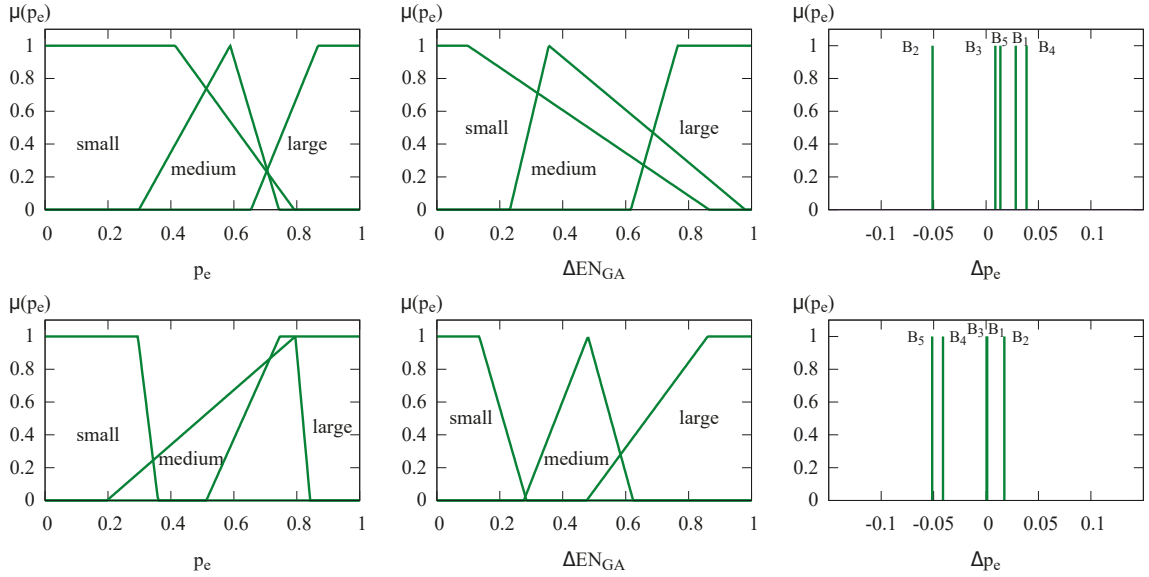

Figure 5. The graphical illustration of membership functions for the fuzzy systems trained by F_3 and F_8 benchmark functions

The presented results show that the combination of F_3 and F_8 strategies, the least often achieved significantly worse results and the most often significantly better. Therefore we used this combination for further experiments.

The Figure 5 depicts the graphical representation of the membership functions for input and output fuzzy sets of the F_3 and F_8 fuzzy systems. The rest of the parameters of these two strategies are presented in Table 3.

The comparison of the results of the proposed FSHPSO-E-AA algorithm that automatically switches between F_3 and F_8 strategies with the FSHPSO-E algorithm, using fuzzy systems F_3 and F_8 separately, is presented in Table 4.

This Table shows that the strategy switching in the proposed FSHPSO-E-AA algorithm for the functions F_1, F_4, F_9, F_{12}, F_{13} allows to obtain much better average results than using these strategies separately. In the case of functions F_1, F_2, F_3, F_5 the best solution was also found definitely closer to the optimal one. In other cases, the solutions were comparable. It should be noted that in the case of function F_{12} the proposed algorithm found the closest solution in each of the 30 repetitions of the experiment.

4.2 Comparison With Other PSO-based Algorithms

In this Section the performance of the FSHPSO-E-AA algorithm is compared with several well-known methods. The algorithms used for comparison are listed as follows:

1. PSO with interia weight (GPSO) [29],
2. Fully informed PSO (FIPSO) [23],
3. Comprehensive learning PSO (CLPSO) [22],
4. Dynamic Tournament Topology PSO (DT-PSO) [35],
5. A Hybrid PSO-GA algorithm (HPSO-GA) [12],
6. A Hybrid of GA and PSO (HGAPSO) [16],

The researchers use different benchmark functions with various configurations (search range and number of dimensions), so in order to compare the results, all the mentioned algorithms have been implemented in C# language. Other hybrid methods, where in one iteration of the PSO algorithm, particles are modified also by genetic operators, were not included in the simulations because their descriptions did not allow for trustworthy implementation. During the simulations their parameters have been set at the values proposed in the corresponding articles and are presented in Table 5.

Table 5. The parameters of algorithms.

Algorithm	Parameters	Source
GPSO	w: 0.75, ψ_1: 1.5, ψ_2: 1.7	[29]
FIPSO	χ: 0.7298, ψ: 4.1	[23]
CLPSO	w_0: 0.9, w_1: 0.4, c: 1.49445, m: 7, T: 2	[22]
DT-PSO	w: 1, ψ: 4.1, K: 0.1, M: 6, P: 0.05	[35]
HPSO-GA	w: 0.828, ψ_1: 1.5, ψ_2: 1.5, p_c: 0.85, p_m: 0.02, γ: 10, β: 15, GA_{NumMax}: 20, GA_{NumMin}: 1, $GA_{MinIter}$: 10, GA_{PsMax}: 20 GA_{PsMin}: 10, $GA_{MaxIter}$: 20	[12]
HGAPSO	χ: 0.8, ψ_1: 1, ψ_2: 1, p_c: 0.8, p_m: 0.1, T: 2	[16]

Table 6. Average time of calculation in seconds for the F_1-F_{15} functions and $D = 30$

Function	GPSO	FIPSO-F	CLPSO	DT-PSO
F_1	0.34	6.61	**0.33**	1.16
F_2	**0.35**	6.58	0.57	1.17
F_3	0.53	6.72	0.82	1.33
F_4	**0.35**	6.64	0.59	1.18
F_5	**0.34**	6.58	0.43	1.17
F_6	0.08	1.12	0.09	**0.04**
F_7	**0.63**	6.59	1.01	1.21
F_8	0.59	6.83	0.98	1.43
F_9	0.57	6.75	0.79	1.36
F_{10}	0.54	6.71	0.54	1.34
F_{11}	0.36	6.78	0.67	**0.22**
F_{12}	0.93	7.07	**0.91**	1.72
F_{13}	0.68	6.83	**0.67**	1.46
F_{14}	0.74	8.22	1.07	2.00
F_{15}	0.52	8.08	1.03	1.41
	HPSO-GA	HGAPSO	FSHPSO-E-AA	
F_1	8.93	1.70	0.39	
F_2	9.01	1.70	0.50	
F_3	9.52	1.76	0.62	
F_4	8.96	1.72	0.42	
F_5	9.01	1.68	0.43	
F_6	9.69	0.41	0.09	
F_7	9.18	1.73	0.77	
F_8	9.77	1.70	0.89	
F_9	9.77	1.76	**0.40**	
F_{10}	9.74	1.77	0.82	
F_{11}	9.77	1.81	0.83	
F_{12}	9.87	1.96	1.40	
F_{13}	9.81	1.84	1.24	
F_{14}	9.73	1.85	**0.48**	
F_{15}	8.18	1.62	**0.37**	

The population size for all PSO variants has been set to $N = 50$ except for the DT-PSO algorithm where Wang et al. [35] recommended set the population size to $N = 60$. For all algorithms the dimensionality of optimized function has been set to $D = 30$ and the maximum number of function evaluations has been set to $FE = 10000 \cdot D$.

Table 7 shows the results obtained by each algorithm. The best value in each row is marked in bold. This Table also reports the p-value obtained in the hypothesis test. In Table 7 the "+" sign means that the compared algorithm is significantly better than FSHPSO-E-AA, the "–" sign that algorithm is significantly worse than FSHPSO-E-AA and "#" that the two algorithms obtain the same results.

Table 7 points out that the proposed algorithm allows to find the best solution for ten benchmark functions and the best average solution for the seven benchmark functions.

Table 6 reports the average time of calculations in seconds. This Table shows that this algorithm is very efficient and its average time is comparable with classical GPSO and CLPSO methods.

The average performance of the algorithms in the function of the number of evaluations is shown in Figure 6.

4.3 The inverted pendulum problem

In this Section we show the performance of the proposed FSHPSO-E-AA method on real-world modelling problem. We choose the inverted pendulum on the cart because it arises as a model of many real systems like Sagway, bicycles, unicycles and rockets (during liftoff). The inverted pendulum is a pendulum that has its center of mass above its pivot point, and it is unstable without additional help. Graphical representation of this model is presented in Figure 7. The model of the inverted pendulum can be defined with the following state variables

Table 7. Comparison of the algorithms for F_1 - F_{15} benchmark functions and $D = 30$

Function		GPSO	FIPSO-F	CLPSO	DT-PSO	HPSO-GA	HGAPSO	FSHPSO-E-AA
F_1	Mean	1.22e-094	4.87e-003	4.22e-024	2.33e-143	5.33e+003	3.79e-005	**3.13e-166**
	Best	3.36e-100	3.38e-033	8.13e-025	1.86e-145	2.51e+003	2.75e-006	**1.98e-323**
	p-Val	*0.000001-	*0.000001-	*0.000001-	*0.000001-	*0.000001-	*0.000001-	
F_2	Mean	1.95e-053	3.12e-002	9.07e-007	**2.26e-070**	3.62e+001	1.26e-003	4.47e-040
	Best	6.05e-060	1.18e-026	7.29e-015	3.91e-071	2.73e+001	5.53e-004	**4.20e-084**
	p-Val	*0.040205-	*0.000001-	*0.000001-	0.516407+	*0.000001-	*0.000001-	
F_3	Mean	1.95e-008	3.36e+003	1.13e+002	2.02e-006	2.64e+004	4.47e-001	**1.17e-019**
	Best	1.28e-009	1.27e+003	7.12e+001	4.64e-008	1.85e+004	8.99e-002	**4.09e-035**
	p-Val	*0.000001-	*0.000001-	*0.000001-	*0.000001-	*0.000001-	*0.000001-	
F_4	Mean	5.19e-004	1.58e+000	5.83e-002	**4.21e-017**	4.30e+001	1.04e-001	1.18e-007
	Best	1.59e-005	4.17e-001	2.53e-002	**1.45e-031**	3.18e+001	7.55e-002	5.86e-010
	p-Val	*0.000001-	*0.000001-	*0.000001-	0.999999+	*0.000001-	*0.000001-	
F_5	Mean	1.34e+001	2.82e+001	2.36e+001	1.35e+001	3.55e+002	1.82e+001	**1.22e+000**
	Best	1.16e-002	2.66e+001	1.74e+001	1.30e+001	1.88e+002	1.49e+001	**2.45e-004**
	p-Val	*0.000001-	*0.000001-	*0.000001-	*0.000001-	*0.000001-	*0.000001-	
F_6	Mean	3.33e-002	1.67e-001	**0.00e+000**	**0.00e+000**	4.75e+003	**0.00e+000**	**0.00e+000**
	Best	**0.00e+000**	**0.00e+000**	**0.00e+000**	**0.00e+000**	2.11e+003	**0.00e+000**	**0.00e+000**
	p-Val	0.500000#	0.050174-	#	#	*0.000001-	#	
F_7	Mean	3.19e-003	2.67e-002	4.36e-003	**1.03e-003**	1.69e+000	2.29e-003	3.56e-003
	Best	1.14e-003	9.45e-003	2.05e-003	**4.03e-004**	8.52e-001	1.08e-003	1.61e-003
	p-Val	0.744791+	*0.000001-	*0.001529-	0.999999+	*0.000001-	0.999333+	
F_8	Mean	3.69e+003	5.16e+003	7.62e+002	3.86e+003	6.87e+003	3.29e+002	**1.34e-002**
	Best	2.74e+003	2.71e+003	2.79e+002	2.17e+003	6.35e+003	5.17e-001	**1.34e-002**
	p-Val	*0.000001-	*0.000001-	*0.000001-	*0.000001-	*0.000001-	*0.000001-	
F_9	Mean	4.38e+001	1.86e+001	9.86e-009	8.59e+000	2.43e+002	2.02e-004	**1.01e-015**
	Best	2.49e+001	1.09e+001	7.50e-013	3.98e+000	2.08e+002	7.93e-006	**0.00e+000**
	p-Val	*0.000001-	*0.000001-	*0.000001-	*0.000001-	*0.000001-	*0.000001-	
F_{10}	Mean	2.90e-001	6.32e-004	5.31e-014	**4.00e-015**	1.31e+001	1.36e-003	8.63e-002
	Best	7.55e-015	**4.00e-015**	2.89e-014	**4.00e-015**	1.08e+001	4.42e-004	8.63e-002
	p-Val	0.919041+	0.999999+	0.999999+	0.999999+	*0.000001-	0.999999+	
F_{11}	Mean	1.83e-002	1.38e-002	**1.17e-013**	3.29e-004	4.48e+001	2.44e-002	3.75e-002
	Best	**0.00e+000**	1.19e-012	**0.00e+000**	**0.00e+000**	2.16e+001	1.51e-005	**0.00e+000**
	p-Val	0.976668+	0.996475+	0.999993+	0.999991+	*0.000001-	0.681919+	
F_{12}	Mean	1.29e-001	6.49e-002	5.94e-027	3.57e-003	7.31e+005	1.10e-007	**1.63e-032**
	Best	**1.62e-032**	4.27e-005	1.45e-027	**1.62e-032**	6.04e+004	1.16e-008	**1.62e-032**
	p-Val	*0.001263-	*0.000001-	*0.000001-	0.500000#	*0.000001-	*0.000001-	
F_{13}	Mean	5.54e-003	1.46e-003	2.17e-024	**1.35e-032**	5.35e+006	2.83e-006	1.85e-003
	Best	**1.35e-032**	4.88e-011	3.40e-025	**1.35e-032**	8.91e+005	1.83e-007	**1.35e-032**
	p-Val	*0.009706-	*0.001167-	*0.001430-	0.998552+	*0.000001-	*0.001430-	
F_{14}	Mean	2.22e+001	1.99e+001	3.33e-002	1.16e+001	2.17e+002	1.34e-004	**2.84e-014**
	Best	5.97e+000	1.10e+001	**0.00e+000**	6.99e+000	1.82e+002	1.01e-005	**0.00e+000**
	p-Val	*0.000001-	*0.000001-	*0.000002-	*0.000001-	*0.000001-	*0.000001-	
F_{15}	Mean	4.06e+002	3.13e+003	3.14e+003	4.73e+002	7.08e+009	2.88e+010	**4.00e+002**
	Best	3.90e+002	4.18e+002	9.56e+002	3.91e+002	2.35e+009	3.93e+002	**3.90e+002**
	p-Val	*0.005705-	*0.000001-	*0.000001-	*0.000400-	*0.000001-	*0.000001-	
Better-Mean		45	28	54	66	1	47	**68**
Sig-Better		60	27	51	72	0	46	**74**
Sig-Worse		46	69	39	19	90	46	**17**

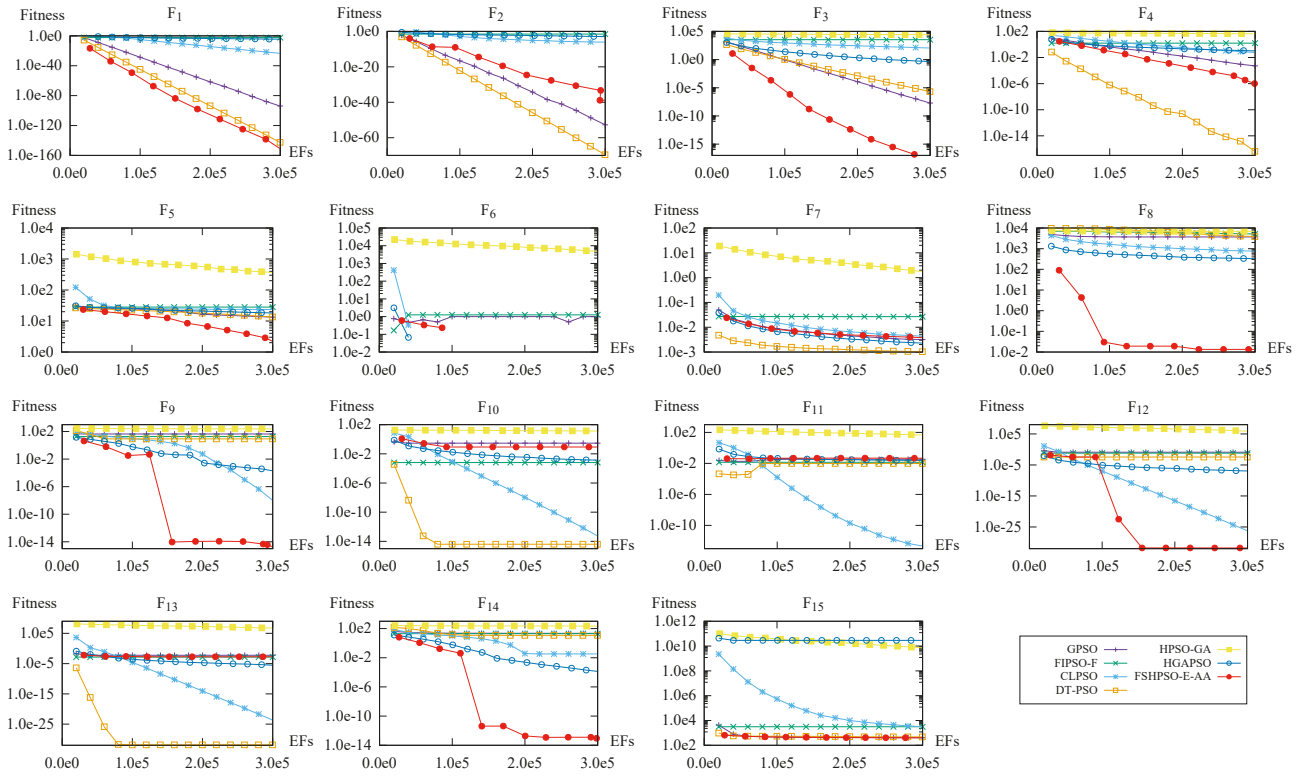

Figure 6. The average performance of the algorithms for F_1 - F_{15} benchmark functions ($D = 30$).

$$x_1 = x, \quad x_2 = \dot{x}, \quad x_3 = \theta, \quad x_4 = \dot{\theta}, \tag{18}$$

where x_1 is the cart position in the direction of the x-axis, x_2 is derivative of movement (velocity), x_3 is the angle of the inverted pendulum from the y-axis (θ), and x_4 is the angular velocity of the pendulum ($\dot{\theta}$).

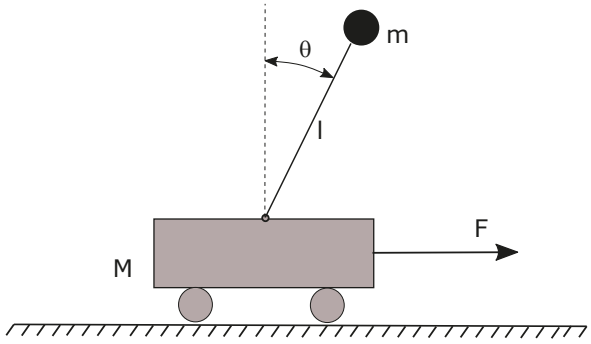

Figure 7. A schematic drawing of the inverted pendulum on a cart

The state transition is described by the formulas (19)-(22)

$$\dot{x}_1 = \dot{x} = x_2, \tag{19}$$

$$\dot{x}_2 = \ddot{x} = \frac{-mg\sin x_3 \cos \mathbf{x}_3 + mlx_4^2 \sin x_3}{M + (1 - \cos^2 x_3)\, m} + \frac{f_\theta m x_4 \cos x_3 + F}{M + (1 - \cos^2 x_3)\, m}, \tag{20}$$

$$\dot{x}_3 = \dot{\theta} = x_4 \tag{21}$$

$$\dot{x}_4 = \ddot{\theta} = \frac{(M+m)\,(g\sin x_3 - f_\theta x_4)}{l \cdot (M + (1 - \cos^2 x_3) \cdot m)} - \frac{\left(lmx_4^2 \sin x_3 + F\right)\cos x_3}{l \cdot (M + (1 - \cos^2 x_3) \cdot m)}, \tag{22}$$

where m means the inverted pendulum mass, M is the mass of the cart, l is the pendulum length, f_θ is the friction in the rotational link, F determines the normal force applied to the cart, and g is the gravity constant.

The MISO Neuro-Fuzzy System (NFS) is used for maintaining the inverted pendulum in the stable pivot point. The NFS is responsible for the determination of the force F based on the x_3 and x_4 state variables. It has five membership functions for each input, which gives in total the 25 fuzzy rules. The conclusion of each rule contains the required

value of force F. The task of the evaluated optimization algorithms is the determination of the parameters of the membership functions for the fuzzy rules. To evaluate solutions we used the following fitness function

$$Fit(\theta, T, \Delta t) = \sum_{t=0}^{T-\Delta t} |\theta(t) \cdot \Delta t|, \qquad (23)$$

where Δt is the integration time.

The parameters of the inverted pendulum model used during the simulations are presented in Table 8.

Table 8. The parameters of the inverted pendulum model

M	m	f_θ	l	θ	F
0.5	0.2	0.1	0.3	± 0.2	± 5

We assume that $T = 5$ and the begging value of $\theta = 0.2$ radians.

As in the experiments described in the previous Section of this article, simulations were repeated 30 times for each optimization algorithm. The results are presented in Table 9.

Table 9. The average results obtained for the inverted pendulum problem in the 30 repetitions.

	Mean	Best	p-Val
GPSO	**0.027**	**0.019**	**0.2552-**
FIPSO-F	0.086	0.038	*0.0001-
CLPSO	0.038	0.026	*0.0001-
DT-PSO	**0.023**	**0.016**	**0.8875+**
HPSO-GA	0.087	0.046	*0.0001-
HGAPSO	0.031	0.021	*0.0189-
FSHPSO-E-AA	**0.026**	**0.018**	

The proposed FSHPSO-E-AA algorithm obtains very similar results as the best one (DT-PSO), however, according to the p-value obtained in the hypothesis test, the differences are statistically insignificant.

Changes of the θ angle during one of the simulations are shown in Figure 8. As can be seen, the FSHPSO-E-AA algorithm obtains the best results - the fastest moving to the stable pivot point without any oscillations.

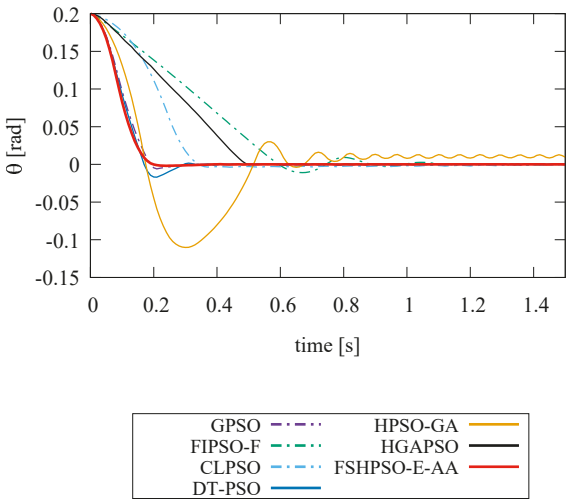

Figure 8. Changes in the θ angle of the inverted pendulum controlled by the NFS trainded by the different algorithms.

5 Conclusions

In this paper, we proposed the incorporating of the auto adaptation mechanism into the FSHPSO-E algorithm. It allows to change the form of fuzzy rules when solving optimization problems. The performance of the proposed FSHPSO-E-AA algorithm was confirmed on the set of benchmark functions and one real-world modelling problem. The presented results show that, our method allows for better performance than the FSHPSO-E algorithm and other analysed algorithms for most benchmark functions.

Acknowledgement

The project financed under the program of the Polish Minister of Science and Higher Education under the name "Regional Initiative of Excellence" in the years 2019 - 2022 project number 020/RID/2018/19, the amount of financing 12,000,000.00 PLN.

Appendix

The detailed description of the problems that have been used in this paper is given below:

1. Sphere problem:

$$\min_{\mathbf{x}} F_1(\mathbf{x}) = \sum_{i=1}^{D} x_i^2,$$

where $\mathbf{x} \in [-100,100]^D$ with known global minimum $F_1(\mathbf{x}^*) = 0$ at $x_i^* = 0$ for $i = 1,\ldots,D$,

2. Schwefel 2.2

$$\min_{\mathbf{x}} F_2(\mathbf{x}) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i|,$$

where $\mathbf{x} \in [-10,10]^D$ with known global minimum $F_2(\mathbf{x}^*) = 0$ at $x_i^* = 0$ for $i = 1,\ldots,D$

3. Schwefel 1.2

$$\min_{\mathbf{x}} F_3(\mathbf{x}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2,$$

where $\mathbf{x} \in [-100,100]^D$ with known global minimum $F_3(\mathbf{x}^*) = 0$ at $x_i^* = 0$ for $i = 1,\ldots,D$

4. Schwefel 2.21

$$\min_{\mathbf{x}} F_4(\mathbf{x}) = \max_i \{|x_i|, 1 \le i \le D\},$$

where $\mathbf{x} \in [-100,100]^D$ with known global minimum $F_4(\mathbf{x}^*) = 0$ at $x_i^* = 0$ for $i = 1,\ldots,D$

5. Rosenbrock

$$\min_{\mathbf{x}} F_5(\mathbf{x}) = \sum_{i=1}^{D-1} \left(100 \left(x_i^2 - x_{i+1} \right)^2 + (x_i - 1)^2 \right),$$

where $\mathbf{x} \in [-2,2]^D$ with known global minimum $F_5(\mathbf{x}^*) = 0$ at $x_i^* = 0$ for $i = 1,\ldots,D$

6. Step Function

$$\min_{\mathbf{x}} F_6(\mathbf{x}) = \sum_{i=1}^{D} \left(\lfloor x_i + 0.5 \rfloor \right)^2,$$

where $\mathbf{x} \in [-100,100]^D$ with known global minimum $F_6(\mathbf{x}^*) = 0$ at $x_i^* = 0$ for $i = 1,\ldots,D$

7. Noise

$$\min_{\mathbf{x}} F_7(\mathbf{x}) = \sum_{i=1}^{D} i x_i^4 + \text{random}[0,1),$$

where $\mathbf{x} \in [-1.28,1.28]^D$ with known global minimum $F_7(\mathbf{x}^*) = 0$ at $x_i^* = 0$ for $i = 1,\ldots,D$

8. Generalized Schwefel 2.26

$$\min_{\mathbf{x}} F_8(\mathbf{x}) = -\sum_{i=1}^{D} (x_i \sin(\sqrt{|x_i|})),$$

where $\mathbf{x} \in [-500,500]^D$ with known global minimum $F_8(\mathbf{x}^*) = -12569.5$ at $x_i^* = 0$ for $i = 1,\ldots,D$

9. Rastrigin

$$\min_{\mathbf{x}} F_9(\mathbf{x}) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10 \right),$$

where $\mathbf{x} \in [-5,5]^D$ with known global minimum $F_9(\mathbf{x}^*) = 0$ at $x_i^* = 0$ for $i = 1,\ldots,D$

10. Ackley

$$\min_{\mathbf{x}} F_{10}(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}} - e^{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)} + 20 + e,$$

where $\mathbf{x} \in [-32,32]^D$ with known global minimum $F_{10}(\mathbf{x}^*) = 0$ at $x_i^* = 0$ for $i = 1,\ldots,D$

11. Griewank

$$\min_{\mathbf{x}} F_{11}(\mathbf{x}) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1,$$

where $\mathbf{x} \in [-600,600]^D$ with known global minimum $F_{11}(\mathbf{x}^*) = 0$ at $x_i^* = 0$ for $i = 1,\ldots,D$

12. Generalized Penalized Function 1

$$\min_{\mathbf{x}} F_{12}(\mathbf{x}) = \frac{\pi}{D} \Bigg\{ 10\sin^2(\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2 \cdot \left[1 + 10\sin^2(\pi y_{i+1}) \right] + (y_D - 1)^2 \Bigg\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4),$$

where $y_i = 1 + 0.25(x_i + 1)$ and $\mathbf{x} \in [-50,50]^D$ with known global minimum $F_{12}(\mathbf{x}^*) = 0$ at $x_i^* = 0$ for $i = 1,\ldots,D$

13. Generalized Penalized Function 2

$$\begin{aligned}\min_{\mathbf{x}} F_{13}(\mathbf{x}) &= 0.1\Big\{ \sin^2(3\pi x_1) \\ &+ \sum_{i=1}^{D-1} (x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right] \\ &+ (x_D-1)^2\left[1+\sin^2(2\pi x_D)\right]\Big\} \\ &+ \sum_{i=1}^{D} u(x_i,10,100,4),\end{aligned}$$

where $\mathbf{x} \in [-50,50]^D$ with known global minimum $F_{13}(\mathbf{x}^*) = 0$ at $x_i^* = 1$ for $i = 1,\ldots,D$

In problems 12 and 13, the penalty function u is given by the following expression

$$u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m, & x_i > a \\ 0, & -a \le \mathbf{x}_i \le a \\ k(-x_i-a)^m, & x_i < -a, \end{cases}$$

14. Shifted Noncontinuous Rastrigin

$$\min_{\mathbf{x}} F_{14}(\mathbf{x}) = \sum_{i=1}^{D} \left(y_i^2 - 10\cos(2\pi y_i) + 10\right),$$

where

$$y_i = \begin{cases} x_i & |x_i| < \frac{1}{2} \\ \frac{\text{round}(2x_i)}{2} & |x_i| \ge \frac{1}{2} \end{cases},$$

and $\mathbf{x} \in [-5,5]^D$ with known global minimum $F_{14}(\mathbf{x}^*) = 0$ at $x_i^* = 0$ for $i = 1,\ldots,D$

15. Shifted Rosenbrock

$$\begin{aligned}\min_{\mathbf{x}} F_{15}(\mathbf{x}) = &\sum_{i=1}^{D-1} \left(100(z_i^2 - z_{i+1})^2 + (z_i-1)^2\right) \\ &+ 390,\end{aligned}$$

where $z_i = x_i - o_i + 1$ and $\mathbf{x} \in [-200,200]^D$ with known global minimum $F_{15}(\mathbf{x}^*) = 390$ at $x_i^* = 0$ for $i = 1,\ldots,D$

References

[1] WF Abd-El-Wahed, AA Mousa, and MA El-Shorbagy. Integrating particle swarm optimization with genetic algorithms for solving nonlinear optimization problems. Journal of Computational and Applied Mathematics, 235(5):1446–1453, 2011.

[2] Mohamed Abdel-Basset, Ahmed E Fakhry, Ibrahim El-Henawy, Tie Qiu, and Arun Kumar Sangaiah. Feature and intensity based medical image registration using particle swarm optimization. Journal of medical systems, 41(12):197, 2017.

[3] Yulian Cao, Han Zhang, Wenfeng Li, Mengchu Zhou, Yu Zhang, and Wanpracha Art Chaovalitwongse. Comprehensive learning particle swarm optimization algorithm with local search for multimodal functions. IEEE Transactions on Evolutionary Computation, 2018.

[4] Kalyanmoy Deb. An introduction to genetic algorithms. Sadhana, 24(4-5):293–315, 1999.

[5] Jinjin Ding, Qunjin Wang, Qian Zhang, Qiubo Ye, and Yuan Ma. A hybrid particle swarm optimization-cuckoo search algorithm and its engineering applications. Mathematical Problems in Engineering, 2019, 2019.

[6] Wenyong Dong and MengChu Zhou. A supervised learning and control method to improve particle swarm optimization algorithms. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47(7):1135–1148, 2016.

[7] Piotr Dziwiński, Łukasz Bartczuk, and Piotr Goetzen. A new hybrid particle swarm optimization and evolutionary algorithm. In Inter. Conf. on Artificial Intelligence and Soft Computing, pages 432–444. Springer, 2019.

[8] P. Dziwiński and Ł. Bartczuk. A new hybrid particle swarm optimization and genetic algorithm method controlled by fuzzy logic. IEEE Transactions on Fuzzy Systems, pages 1–1, 2019.

[9] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pages 39–43. Ieee, 1995.

[10] Shu-Kai S Fan, Yun-Chia Liang, and Erwie Zahara. A genetic algorithm and a particle swarm optimizer hybridized with nelder–mead simplex search. Computers & industrial engineering, 50(4):401–425, 2006.

[11] Shu-Kai S Fan and Erwie Zahara. A hybrid simplex search and particle swarm optimization for unconstrained optimization. European Journal of Operational Research, 181(2):527–548, 2007.

[12] Harish Garg. A hybrid pso-ga algorithm for constrained optimization problems. Applied Math. and Comput., 274:292–305, 2016.

[13] David E Goldberg. Genetic algorithms. Pearson Education India, 2006.

[14] Yue-Jiao Gong, Jing-Jing Li, Yicong Zhou, Yun Li, Henry Shu-Hung Chung, Yu-Hui Shi, and Jun Zhang. Genetic learning particle swarm optimization. IEEE transactions on cybernetics, 46(10):2277–2290, 2015.

[15] J-SR Jang and Chuen-Tsai Sun. Neuro-fuzzy modeling and control. Proceedings of the IEEE, 83(3):378–406, 1995.

[16] Chia-Feng Juang. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Trans. on Systems, Man, and Cybernetics, 34(2):997–1006, 2004.

[17] Yi-Tung Kao and Erwie Zahara. A hybrid genetic algorithm and particle swarm optimization for multimodal functions. Applied Soft Computing, 8(2):849–857, 2008.

[18] Oliver Kramer. Genetic algorithm essentials, volume 679. Springer, 2017.

[19] Evan Krell, Alaa Sheta, Arun Prassanth Ramaswamy Balasubramanian, and Scott A. King. Collision-free autonomous robot navigation in unknown environments utilizing pso for path planning. Journal of Artificial Intelligence and Soft Computing Research, 9(4):267–282, 2019.

[20] RJ Kuo and YS Han. A hybrid of genetic algorithm and particle swarm optimization for solving bi-level linear programming problem–a case study on supply chain model. Applied Mathematical Modelling, 35(8):3905–3917, 2011.

[21] H. Li and L. Li. A novel hybrid particle swarm optimization algorithm combined with harmony search for high dimensional optimization problems. In The 2007 International Conference on Intelligent Pervasive Computing (IPC 2007), pages 94–97, Oct 2007.

[22] Jing J Liang, A Kai Qin, Ponnuthurai N Suganthan, and S Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE transactions on evolutionary computation, 10(3):281–295, 2006.

[23] Rui Mendes, James Kennedy, and José Neves. The fully informed particle swarm: simpler, maybe better. IEEE transactions on evolutionary computation, 8(3):204–210, 2004.

[24] Tien-Szu Pan, Thi-Kien Dao, Shu-Chuan Chu, et al. Hybrid particle swarm optimization with bat algorithm. In Genetic and evolutionary computing, pages 37–47. Springer, 2015.

[25] K Premalatha and AM Natarajan. Hybrid pso and ga for global maximization. Int. J. Open Problems Compt. Math, 2(4):597–608, 2009.

[26] J. Robinson, S. Sinton, and Y. Rahmat-Samii. Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna. In IEEE Antennas and Propagation Society International Symposium (IEEE Cat. No.02CH37313), volume 1, pages 314–317 vol.1, June 2002.

[27] Leszek Rutkowski. Computational intelligence: methods and techniques. Springer Science & Business Media, 2008.

[28] XH Shi, YC Liang, HP Lee, Chun Lu, and LM Wang. An improved ga and a novel pso-ga-based hybrid algorithm. Information Processing Letters, 93(5):255–261, 2005.

[29] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In 1998 IEEE Inter. Conf. on Evolut. Comput. Proc., pages 69–73. IEEE, 1998.

[30] Narinder Singh and SB Singh. Hybrid algorithm of particle swarm optimization and grey wolf optimizer for improving convergence performance. Journal of Applied Mathematics, 2017, 2017.

[31] SN Sivanandam and SN Deepa. Genetic algorithms. In Introduction to genetic algorithms, pages 15–37. Springer, 2008.

[32] George Tambouratzis. Using particle swarm optimization to accurately identify syntactic phrases in free text. Journal of Artificial Intelligence and Soft Computing Research, 8(1):63–67, 2018.

[33] Jianchao Tang, Guoji Zhang, Binbin Lin, and Bixi Zhang. A hybrid pso/ga algorithm for job shop scheduling problem. In International Conference in Swarm Intelligence, pages 566–573. Springer, 2010.

[34] F. Valdez, P. Melin, O. Castillo, and O. Montiel. A new evolutionary method with a hybrid approach combining particle swarm optimization and genetic algorithms using fuzzy logic for decision making. In 2008 IEEE Congress on Evolutionary Computation, pages 1333–1339, June 2008.

[35] Lin Wang, Bo Yang, and Jeff Orchard. Particle swarm optimization using dynamic tournament topology. Applied Soft Computing, 48:584–596, 2016.

[36] Xi-Huai Wang and Jun-Jun Li. Hybrid particle swarm optimization with simulated annealing. In Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 04EX826), volume 4, pages 2402–2405. IEEE, 2004.

[37] Xin Yao, Yong Liu, and Guangming Lin. Evolutionary programming made faster. IEEE Transactions on Evolutionary computation, 3(2):82–102, 1999.

[38] Fengchun Zhang, Wei Fan, Xingfeng Wu, and Gert F Pedersen. Performance testing of mimo device with the wireless cable method based on particle swarm optimization algorithm. In 2018 International Workshop on Antenna Technology (iWAT), pages 1–4. IEEE, 2018.

Piotr Dziwiński received the M.Sc. degree in 2003 and the Ph.D. degree in 2008 in computer science, both from the Czestochowa University of Technology. Since 2008 he has been an Assistant Professor at the Institute of Computational Intelligence, Czestochowa University of Technology, Czestochowa, Poland. He has published 17 technical papers in conference proceedings and book chapters. His main fields of interests include swarm intelligence, evolutionary algorithms, hybrid methods, and fuzzy systems.

Łukasz Bartczuk received the M.Sc. degree in 2003 and the Ph.D. degree in 2008 in computer science, both from the Czestochowa University of Technology. Since 2008 he has been an Assistant Professor at the Institute of Computational Intelligence, Czestochowa University of Technology, Czestochowa, Poland. He has published about 20 technical papers in conference proceedings and book chapters. His main fields of interests include fuzzy systems, evolutionary algorithms, and hybrid methods.

Józef Paszkowski graduated from the Military University of Technology in Warsaw, Faculty of Cybernetics (M.Sc. 1976) with specialization in organization and design of IT systems. He completed post-graduate studies in management systems at the University of Lodz and programming technology at the University of Warsaw/ IMM PAN in Warsaw. In 1995 he received the Ph.D. in the field of medical informatics. In the years 2000 - 2004 he worked as a specialist in an IT company. Currently, he is an Assistant Professor at the University of Social Sciences in Łódź. He specializes in analyzing and modeling data processing systems for distributed business and public utilities. His research and teaching interests include database systems, process management methods, information systems design technologies, computational intelligence, and knowledge engineering methods.