On the Application of a Hybrid Genetic-Firework Algorithm for Controllers Structure and Parameters Selection

Krystian Łapa and Krzysztof Cpałka

Abstract An approach proposed in this paper uses a new hybrid population-based algorithm. This algorithm is a fusion between genetic algorithm and firework algorithm. Proposed approach aims on solving complex optimization problems in which not only structure parameters of the solution have to be selected, but also the mentioned structure. Proposed approach is based on multiple linear correction terms PID connected using proposed dynamic structure. In simulations a problem of selecting structure and its parameters for automatic control was used. For system evaluation a weighted multi-objective fitness function was used, which can consider elements connected to the simulation problems taken into consideration, such as: RMSE error, oscillations of the controller output signal, controller complexity and overshoot of the control signal.

Keywords Hybrid population-based algorithm · Selecting structure · Controller

1 Introduction

Population-based algorithms belong to heuristic algorithms and they are used mostly for solving optimization problems. They are different than traditional optimization methods because: (a) they do not process parameters of the problem, but encoded version of these parameters, (b) they search problem parameters not on the basis of one solution but they use a population of solutions, (c) they use objective function directly, not its derivatives, (d) they use probabilistic, not deterministic

K. Łapa (✉) · K. Cpałka
Institute of Computational Intelligence, Częstochowa University of Technology, Częstochowa, Poland
e-mail: krystian.lapa@iisi.pcz.pl

K. Cpałka
e-mail: krzysztof.cpalka@iisi.pcz.pl

L. Borzemski et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part I*, Advances in Intelligent Systems and Computing 429, DOI 10.1007/978-3-319-28555-9_10

selection of rules. Due to that they have an advantage in comparison with other kind of approaches such as analytical methods, randomization methods, etc. (see e.g. [4, 10]).

Among population-based algorithms a different approaches can be found (see Fig. 1): inspired by nature, inspired by ecology (e.g. Biogeography-Based Optimization, see [11]) or based on social evolution (e.g. Imperialist Competitive Algorithm, see [1]) etc. It is worth to mention that all population-based algorithms benefit from the evolutionary principle of survival of individuals (solutions of problem specially encoded by parameters) and use mechanisms of exploration and exploitation (defined by specified algorithm) of search space to improve fitness of individuals. The aim of population-based algorithms is to minimize or maximize values of fitness function adopted to the considered problem. Algorithms stop working when a stop condition is achieved (e.g. when value of fitness function reaches assumed value).

Most of the presented in the literature population-based algorithms cannot be used directly to solve a group of complex optimization problems (which are important from a practical point of view). Those problems concern to find both the structure and the structure parameters of solution (which is needed e.g. in neural networks, fuzzy systems, biometric systems and controllers—considered in this paper) [3, 7, 14]. Majority of population-based algorithms are focused on searching parameters of structures defined by user. For more elastic solutions (considering searching both the structure and the structure parameters) hybrid population-based algorithms can be used. In this type of algorithms the structure of the problem can be encoded for example in binary parameters and these parameters can be tuned using genetic algorithm operators. The parameters of the structure can be tuned using any population-based algorithm designed for optimization. The development of hybrid algorithm requires a proper synchronization of its algorithms, components and a proper balance between exploration and exploitation of searching space ensured.

In this paper a new hybrid population-based algorithm is presented. It is based on fusion between firework algorithm (see e.g. [13]) and genetic algorithm (see e.g. [2]). This algorithm was tested on selection of the structure and the structure parameters for controller based on linear correction terms (a controller with an object are a control system). In the literature many approaches for automatization of

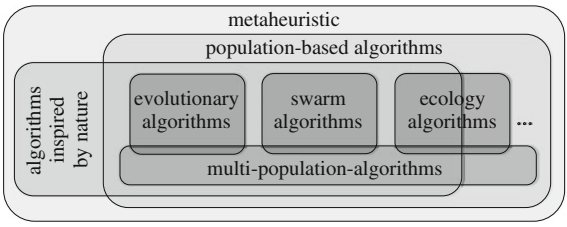

Fig. 1 Division of the population-based algorithms

this process can be found (see e.g. [5]). However, they are based mostly on selecting parameters of controllers with structures experimentally specified by experts. Thus, new and original element of the paper is (a) the proposed hybrid algorithm and (b) a way of its use for the automatic selection of the structure and parameters of the controller in automatic control system.

This paper is organized into four sections. Section 2 presents a description of problem of selecting the structure and structure parameters of controllers. In Sect. 3 a proposed hybrid population-based algorithm is presented. In Sect. 4 simulation results are drawn. Conclusions are presented in Sect. 5.

2 Description of Problem of Selecting the Structure and Structure Parameters of Controllers

In this paper a problem of selecting the structure and structure parameters based on linear correction terms is considered. Linear correction terms correspond to the needs of most automation systems (see e.g. [8]) and they are the most frequently used in practice (see e.g. [9]). Controllers which base on linear correction terms can consist of many Control Blocks (CB) (Fig. 2a). Each of CB can consist of correction terms such as: proportional (P), differential (I) and derivative (D). Moreover, each correction term consist of real number parameter: for P it is a reinforcement parameter K_p, for I it is a time constant T_i, for D it is a time constant T_d. An adequate cooperation of CB (including proper structure of the controller) should be ensured to achieve a proper quality of control.

Proposed method is based on the idea of using general structure of controller which can be modified during training process (it can be reduced or revived). The process of modification of the structure aims to obtain the most simple structure (thanks to properly defined fitness function considering, among others, complexity of the solution), which possibly best suits the control criteria. The main (general) structure of controller of MISO type (multiple input, single output) is presented in Fig. 3. It consists of a subset of selected CB blocks presented in Fig. 2a. The control signal from the block CB may be described as follows:

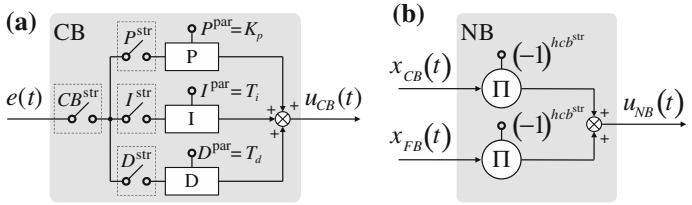

Fig. 2 Main structure of: **a** control block (CB) and **b** node block (NB)

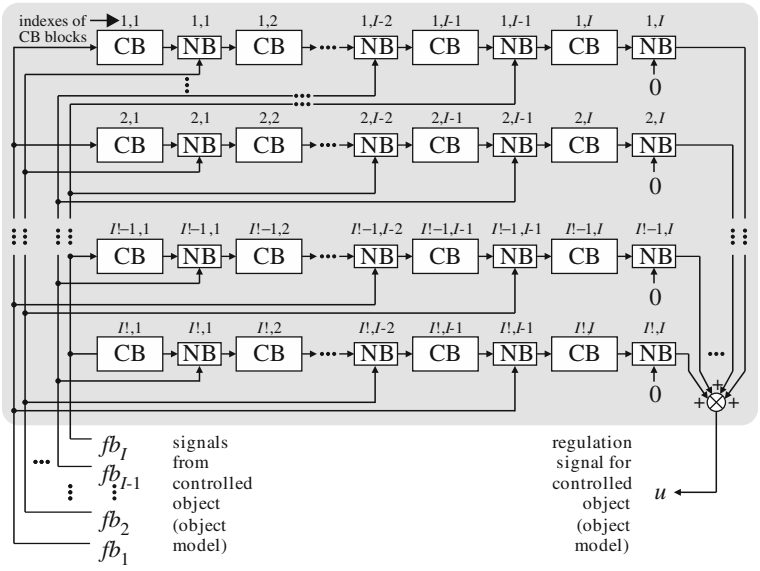

Fig. 3 The generalized structure of the controller based on a linear correction terms, designed by automatically evolutionary reduction

$$u_{CB}(t) = P^{\text{str}} \cdot K_p \cdot e(t) + I^{\text{str}} \cdot \frac{1}{T_i} \cdot \int_0^t e(t)dt + D^{\text{str}} \cdot T_d \cdot \frac{de(t)}{dt}, \tag{1}$$

where $e(t)$ stands for input signal attached to CB block, $u_{CB}(t)$ stands for CB block output signal. Each correction term in (1) is marked symbolically by *key* $P^{\text{str}} \in \{0, 1\}$, $I^{\text{str}} \in \{0, 1\}$ or $D^{\text{str}} \in \{0, 1\}$. The status of the *keys* corresponds to the occurrence of a merger or a break in the circuit (see Fig. 3). In the proposed method the key values are selected evolutionarily. Moreover, structure of the controller also contains Node Blocks (NB) (Fig. 2b) described as follows:

$$u_{NB}(t) = (-1)^{hcb^{\text{str}}} \cdot x_{CB}(t) + (-1)^{hfb^{\text{str}}} \cdot x_{fb}(t), \tag{2}$$

where is a type of feedback of signal connected to the output of the corresponding block CB (if then it is a positive feedback, if then it is a negative feedback) and a type of feedback of signal connected to the corresponding input of controller structure.

The aim of the proposed method is to select structure and parameters related to the structure. Selection of the structure is based on modification (reduction/addition) of correction terms and selection of feedbacks occurring in the controller. The selection process promotes these solutions, in which the number of attached keys is as small as possible.

3 Hybrid Genetic-Firework Population-Based Algorithm

Proposed hybrid genetic-firework population-based algorithm is a fusion between genetic algorithm and firework algorithm. The aim of genetic algorithm part is to select the structure of the controller, the aim of firework algorithm is to select the parameters of the controller. Both algorithms work simultaneously. The well-known idea of genetic algorithm is based on evolution of species (see e.g. [2]), the idea of firework algorithm is based on the behaviour of fireworks and their sparks (see e.g. [13]). In the firework algorithm each *firework* and *spark* are individuals representing single solutions for considered problem. Exploding of *firework* generates specified number of *sparks* around it and covers considered search space. After creation of *sparks*, all individuals are evaluated and best solutions are selected for next step of the algorithm (these solutions became new *fireworks*). Repeating such actions a certain number of times (resulting from the assumed number of steps of the algorithm) gives a real chance to get close to the optimal solution for the considered problem.

The next part of this section describes: **(a)** encoding method for *fireworks* and *sparks* (Sect. 3.1), **(b)** evaluation method for *fireworks* and *sparks* (Sect. 3.2) and **(c)** evolution of the *fireworks* and *sparks* (Sect. 3.3).

3.1 Encoding of the Structure and Parameters

The solutions encoded in population are marked as $\mathbf{X}_j$, $j = 1, \ldots, N$, where N stands for the number of individuals in population (each individual represent single controller). Each individual consists of two parts of parameters: $\mathbf{X}_j^{\text{str}}$ and $\mathbf{X}_j^{\text{par}}$ $\left(\mathbf{X}_j = \left\{\mathbf{X}_j^{\text{str}}, \mathbf{X}_j^{\text{par}}\right\}\right)$. The part $\mathbf{X}_j^{\text{str}}$ is used to encode structure of the controller, and it is expressed as follows::

$$\mathbf{X}_j^{\text{str}} = \begin{bmatrix} CB_{j,1,1}^{\text{str}}, P_{j,1,1}^{\text{str}}, I_{j,1,1}^{\text{str}}, D_{j,1,1}^{\text{str}}, hcb_{j,1,1}^{\text{str}}, hfb_{j,1,1}^{\text{str}}, \ldots, \\ CB_{j,1,I}^{\text{str}}, P_{j,1,I}^{\text{str}}, I_{j,1,I}^{\text{str}}, D_{j,1,I}^{\text{str}}, hcb_{j,1,I}^{\text{str}}, \ldots, \\ CB_{j,I!,1}^{\text{str}}, P_{j,I!,1}^{\text{str}}, I_{j,I!,1}^{\text{str}}, D_{j,I!,1}^{\text{str}}, hcb_{j,I!,1}^{\text{str}}, hfb_{j,I!,1}^{\text{str}}, \ldots, \\ CB_{j,I!,I}^{\text{str}}, P_{j,I!,I}^{\text{str}}, I_{j,I!,I}^{\text{str}}, D_{j,I!,I}^{\text{str}}, hcb_{j,I!,I}^{\text{str}} \end{bmatrix} = \left[X_{j,1}^{\text{str}}, \ldots, X_{j,L^{\text{str}}}^{\text{str}}\right], \quad (3)$$

where each parameter $\mathbf{X}_{j,g}^{\text{str}}$, $g = 1, \ldots, L^{\text{str}}$, encodes information about corresponding *key* (CB^{str}, P^{str}, I^{str}, D^{str}, hcb^{str} or hfb^{str}) of controller structure, $L^{\text{str}} = 6 \cdot I! \cdot I - I!$ stands for the number of parameters of the individual $\mathbf{X}_j^{\text{str}}$ (in the practice controllers use small amount of inputs which translate into processable L^{str}). The part $\mathbf{X}_j^{\text{par}}$ encodes parameters of controller, and it is defined as follows:

$$\mathbf{X}_j^{\text{par}} = \begin{bmatrix} P_{j,1,1}^{\text{par}}, I_{j,1,1}^{\text{par}}, D_{j,1,1}^{\text{par}}, \ldots, \\ P_{j,1,I}^{\text{par}}, I_{j,1,I}^{\text{par}}, D_{j,1,I}^{\text{par}}, \ldots \\ P_{j,I!,1}^{\text{par}}, I_{j,I!,1}^{\text{par}}, D_{j,I!,1}^{\text{par}}, \ldots, \\ P_{j,I!,I}^{\text{par}}, I_{j,I!,I}^{\text{par}}, D_{j,I!,I}^{\text{par}} \end{bmatrix} = \left[X_{j,1}^{\text{par}}, \ldots, X_{j,L^{\text{par}}}^{\text{par}} \right], \tag{4}$$

where each parameter $X_{j,g}^{\text{par}}$, $g = 1, \ldots, L^{\text{par}}$, encodes information about real number parameter K_p, T_i or T_d of a CB block of the controller, $L^{\text{par}} = 3 \cdot I! \cdot I$ stands for the number of parameters of individual $\mathbf{X}_j^{\text{par}}$.

3.2 Individuals Evaluation

The way of defining fitness function in most of the control systems does not depend on algorithm but on considered problem. In case of selecting structure and parameters of controller, fitness function can consist the following elements: RMSE error, oscillations of the controller output signal, controller complexity and overshoot of the control signal. This is a very important issue, because e.g. **(a)** High number of the controller output signal oscillations tends to induce an excessive use of mechanical control parts and may cause often big changes of the controller output signal value. **(b)** The overshoot of the control signal is not acceptable in many industrial applications. The individual evaluation function (maximization problem) is described as follows:

$$ff\left(\mathbf{X}_j\right) = \left(RMSE_j + c_j \cdot w_c + os_j \cdot w_{os} + ov_j \cdot w_{ov}\right)^{-1} \tag{5}$$

where $w_c \in [0, 1]$ denotes a weight factor for the complexity of the controller structure, $c_j > 0$ denotes the complexity of the controller structure described by the formula:

$$c_j = \sum_{g=1}^{L^{\text{str}}} X_{j,g}^{\text{str}}, \tag{6}$$

$w_{ov} \in [0, 1]$ denotes a weight for the overshoot factor, $ov_j \geq 0$ denotes value of the greatest overshoot of the controlled s^1 signal, $w_{os} \in [0, 1]$ denotes a weight for the oscillations factor and finally $os_j \geq 0$ denotes oscillation count of controller output signal calculated as follows:

$$os_j = \sum_{o=1}^{O-1} |r_o - r_{o+1}|, \tag{7}$$

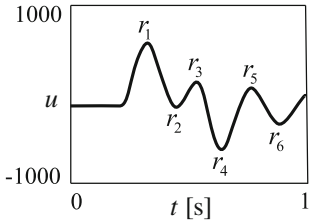

Fig. 4 Minima and maxima of the output signal u

where r_o are (sorted by time value) minima and maxima of the signal u according to Fig. 4. It is worth noting that the function of the form (7) should also count the oscillations with minimum amplitude as an undesirable phenomenon in control systems.

3.3 Evolution Process

The hybrid genetic-firework algorithm proposed in this paper works according to the following steps:

Step 1. Initialization of *fireworks* $\mathbf{X}_j$, $j = 1, \ldots, N$. Individuals in this algorithm are called *fireworks* and *sparks*. Each parameter of *firework* $\mathbf{X}_j^{\text{str}}$ (Sect. 3.1) is chosen randomly from the set $\{0, 1\}$, each parameter $\mathbf{X}_j^{\text{par}}$ (coding parameters of controllers) is generated randomly from ranges related to the considered problem.

Step 2. Evaluation of the initialized population. Each *firework* is evaluated by the fitness function defined in Sect. 3.2.

Step 3. Exploding of the *fireworks*. Each *firework* explodes and generates specified number of *sparks*. This number is calculated for each individual on the basis of fitness function value of *fireworks* (thanks to that more fittest *fireworks* get more *sparks* and vice versa):

$$\hat{s}_j = m \cdot \frac{y_{\max} - ff(\mathbf{X}_j) + \check{n}}{\sum_{k=1}^{n} (y_{\max} - ff(\mathbf{X}_k)) + \check{n}}, \tag{8}$$

where m is a parameter controlling total number of *sparks*, $y_{\max}$ is a best value of fitness function of *fireworks*, $\check{n}$ is a smallest constant in the computer (which prevents division by zero). This number ($\hat{s}_j$) is then reduced (projected) to the range $[b \cdot m, a \cdot m]$, where parameters of the algorithm a and b should satisfy the condition $a < b < 1$. Limiting the number of sparks is performed in order to: (a) prevent the domination of the entire population by fireworks outstanding good value of the evaluation function, (b) allocate sparks even to the fireworks which in the current step have a bad value of evaluation function. The locations of *sparks* are obtained

by mimicking the *firework* explosion process. Each *spark* is a clone of *firework* (it gets the same parameters and the same structure as *firework*) with specified number of parameters modified randomly in a calculated range (range stands 'amplitude of explosion' and it is calculated individually for each *firework*). The value of amplitude of explosion for individual $\mathbf{X_j}$ depends on the fitness function value:

$$A_j = \hat{A} \cdot \frac{ff(\mathbf{X}_j) - y_{\min} + \check{n}}{\sum_{k=1}^{n} (ff(\mathbf{X}_k) - y_{\min}) + \check{n}}, \tag{9}$$

where $\hat{A}$ is a parameter controlling maximum range of *sparks*, $y_{\min}$ is the worst value of fitness function of *fireworks*. Thanks to that, fittest *fireworks* generate *sparks* close to them (exploitation) and vice versa (exploration). In this step additional number of *sparks* is generated on the basis of randomly chosen *fireworks* with modification of specified number of parameters using Gaussian explosion (to maintain diversity of population).

Step 4. Structure modification. In this step a structure of *sparks* generated in Step 3 is modified by mutation operator (known from genetic algorithm). For each *spark* a number from unit interval is generated randomly. If this number is smaller than mutation probability $p_m \in (0, 1)$, *spark* structure will be modified as follows: for each structure parameter a number from unit interval is generated randomly, if this number is smaller than mutation probability, value of the structure parameter is changed to the opposite value (from 0 to 1 and vice versa).

Step 5. Evaluation of *sparks*. In this step each *spark* generated in Step 3 and modified in Step 4 is evaluated by fitness function defined in Sect. 3.2.

Step 6. Selection of new population. New population obtains one of the actually best *firework* and $N - 1$ *sparks* chosen from *sparks* generated in Step 3 and modified in Step 4. In the original version of the algorithm process of selecting individuals takes into account only their diversity: which gives more chances to choose (by the method of roulette wheel) for those individuals with greater distance from the others. It promotes a fuller exploration of search space, however, can lead to degeneration of the population. For this reason, the proposed approach also takes into account the value of fitness function. Thus, the probability of selecting an individual $\mathbf{X}_j$ is defined as follows:

$$p(\mathbf{X}_j) = \frac{\sum_{k=1}^{K} \|\mathbf{X}_j - \mathbf{X}_k\|}{ff(\mathbf{X}_j)}. \tag{10}$$

Step 7. Replacement of the old population by population generated in the previous step. All individuals from new population are treated as *fireworks*. In this step a stop condition is checked. This condition affects the number of iterations of the algorithm. If this condition is not met, then algorithm goes back to Step 3.

Detailed information on the used algorithms can be found e.g. in [2, 13].

4 Simulations Results

In our simulations a problem of designing controller structure and tuning parameters for double spring-mass-damp object was considered (see Fig. 5). The purpose of the controller was the generation of such a control signal (acting on masses and), to adjust in the best way a position of the mass to the reference position. The motion equations for the mass m_1 (for position s^1, velocity v^1 and acceleration a^1) are described as follows:

$$s_n^1 = s_{n-1}^1 + v_{n-1}^1 \cdot T + \left(a_{n-1}^1 \cdot T^2\right) \cdot 0.5, \tag{11}$$

$$v_n^1 = v_{n-1}^1 + a_{n-1}^1 \cdot T, \tag{12}$$

$$a_n^1 = \left(\left(s_n^2 - s_n^1\right) \cdot k - v_n^1 \cdot \mu\right) \cdot m_1^{-1}, \tag{13}$$

where n and $n - 1$ denotes current and previous simulation step respectively, k is spring constant. Analogically, for mass m_2, the motion equations (for position s^2, velocity v^2 and acceleration a^2) have the following form:

$$s_n^2 = s_{n-1}^2 + v_{n-1}^2 \cdot T + \left(a_{n-1}^2 \cdot T^2\right) \cdot 0.5, \tag{14}$$

$$v_n^2 = v_{n-1}^2 + a_{n-1}^2 \cdot T, \tag{15}$$

$$a_n^2 = \left(\left(u - s_n^2\right) \cdot k - v_n^2 \cdot \mu\right) \cdot m_1^{-1}, \tag{16}$$

where u is controller output signal and μ is coefficient of kinetic friction. Remarks about considered model can be summarized as follows: **(a)** Object parameters values were set as follows: spring constant k was set to 10 N/m, coefficient of friction $\mu = 0.5$, masses $m_1 = m_2 = 0.2$ kg. Initial values of: s^1, v^1, s^2 and v^2 were set to zero (which means that the masses were in the rest state at their initial positions). **(b)** Signals fb_i used in structure were set to: s^1, s^2, $s^* - s^1$ respectively. **(c)** Simulation length was set to 10 s, a shape of the reference signal s^* (trapezoid) is presented in Fig. 6, a shape of test signal s^* (sinuous) is also presented in Fig. 6. **(d)** Search range for genes encoding controller parameters were set as follows: $P = [0, 20]$, $I = [0, 50]$, $D = [0, 5]$. **(e)** Output signal of the controller was limited to the range $u \in (-2, +2)$. **(f)** Quantization resolution for the output signal u of the controller as well as for the position sensor for s^1 and s^2 was set to 10 bit. **(g)** Time

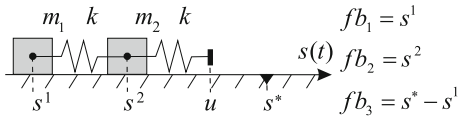

Fig. 5 Simulated spring-mass-damp object

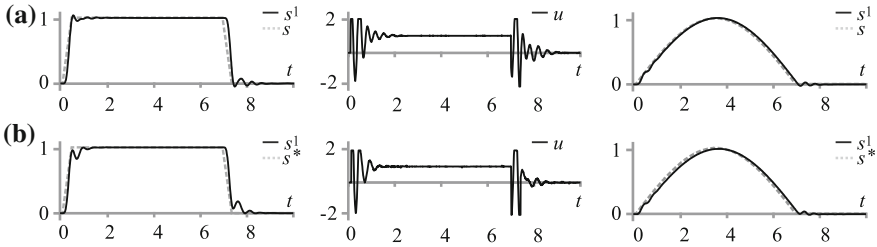

Fig. 6 Signal values s^1, s^* and output signal u for case: **a** FAGA-1, **b** FAGA-2

step in the simulation was equal to $T = 0.1$ ms, while interval between subsequent controller activations were set to twenty simulation steps.

The authorial environment (implemented in C++) was used for simulations. Parameters of the algorithms for the simulations were determined as follows: the number of individual (*fireworks*) N was set to 10, the number of *sparks* was set to $m = 100$ [13], the number of additional *sparks* was set to 10, bounds parameters for number of *sparks*: $a = 0.04$ and $b = 0.80$, maximum amplitude of explosion was set to $\hat{A} = 0.5$, the algorithm performs 1000 steps (generations), the mutation probability was set as $p_m = 0.3$. In our simulations, RMSE error function of the individual was described by the following formula:

$$RMSE_j = \sqrt{\frac{1}{Z}\sum_{n=1}^{Z}\varepsilon_{j,n}^2} = \sqrt{\frac{1}{Z}\sum_{n=1}^{Z}\left(s_{j,n}^* - s_{j,n}^1\right)^2}, \tag{17}$$

where $n = 1, \ldots, Z$, denotes sample index, Z denotes the number of samples, $\varepsilon_{j,n}$ denotes controller tracking error for the sample $s_{j,n}^*$ denotes the value of the reference signal of the controlled value for the sample n, $s_{j,n}^1$ denotes its current value for the sample n. Moreover, the following weights of fitness function components were used: $w_{os} = 0.0100$ and $w_{ov} = 0.0001$. At the same time two cases associated with different weight values w_c were considered:

Case 1. This case (marked further as FAGA-1) aims to obtain high accuracy of the control system: $w_c = 0.0010$.

Case 2. This case (marked further as FAGA-2) aims to obtain low complexity of the control system: $w_c = 0.0040$.

The conclusions of the simulations can be summarized as follows:

– The controllers obtained for considered in simulations problem using hybrid genetic-firework algorithms perform well (see Fig. 7): the level of oscillations (in case FAGA-1 a best value for oscillations was achieved—see Table 2) and overshooting is low, the accuracy of the system is more than satisfactory. This applies to situations, where testing of the controller was made using signal s^* with trapezoidal shape (used in a training phase) and also when signal s^* have sinusoidal shape (used only in test phase).

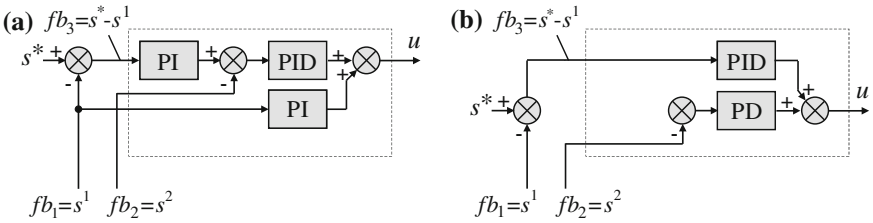

Fig. 7 Structure of the controller obtained for case: **a** FAGA-1, **b** FAGA-2 (structures denoted by *dashed line* were formed by reduction of the structure shown in Fig. 3)

– The structure of controllers for both cases considered in simulations was chosen according to assumptions. For case FAGA-1 the structure is more complex than for case FAGA-2 (see Fig. 7), but it works more accurate (see Fig. 6a, b). It is worth to mention that, in our previous work (see [12]) different methods for automatic selection the structure and the structure parameters were used, however we did not achieved as simplest structure as in case FAGA-2 (see Table 1) with comparable quality of control (see Table 2).
– In our previous works problem of selecting both the structure and the structure parameters was considered (see [6, 12]). Those papers contains methods based on simple algorithms (e.g. genetic algorithm, evolutionary strategy (μ, λ)) and on standard population-based algorithms (e.g. gravitation algorithm, firefly algorithm). Results obtained in this paper differ than results from other papers (see Tables 1 and 2). Accuracy obtained for case FAGA-1 is close to the best accuracy achieved in previous researches (see Table 2). At the same time, the structure obtained in case FAGA-2 is the simplest (in comparison to other results) and preforms appropriate well.

Table 1 Number of correction terms (*nterms*) for best individuals of population

nterms	FAGA-1	FAGA-2	Our previous results [12]
P	3	**2**	2–5
I	3	**1**	1–3
D	**1**	2	1–3
All	7	**5**	6–10

Table 2 Values of the fitness function (5) for best individuals of population

Name	FAGA-1	FAGA-2	Our previous results [12]
RMSE	0.0547	0.0623	0.0502–0.1790
$c_j \cdot w_c$	0.0070	0.0200	0.0060–0.0100
$os_j \cdot w_{os}$	0.0101	0.0113	0.0123–0.0350
$ov_j \cdot w_{ov}$	0.0004	0.0001	0.0001–0.0014
ff	13.8504	10.6723	4.9140–13.9082

5 Conclusions

In this paper a new hybrid genetic-firework algorithm is proposed. This algorithm can be used to solve complex optimization problems in which: both the structure and the structure parameters of the controller have to be found and various criteria for selection have to be taken in consideration (e.g. related to the complexity, accuracy, etc.). The algorithm was used for automatic selection of the controller, but can also be used e.g. for selection of structure and structure parameters of another kind of computational intelligence algorithm (e.g. neural network, decision tree, neuro-fuzzy systems, etc.). Results received in the simulations are positive and obtained structures of the controller are simple.

Acknowledgments The authors would like to thank the reviewers for very helpful suggestions and comments in the revision process.

The project was financed by the National Science Centre (Poland) on the basis of the decision number DEC-2012/05/B/ST7/02138.

References

1. Atashpaz-Gargari, E., Lucas, C.: Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. IEEE Cong. Evol. Comput. **7**, 4661–4666 (2007)
2. Binitha, S., Siva, S.S.: A survey of bio-inspired optimization algorithms. Int. J. Soft Comput. Eng. (IJSCE) **2**(2) (2012)
3. Cpałka, K., Łapa, K., Przybył, A., Zalasiński, M.: A new method for designing neuro-fuzzy systems for nonlinear modelling with interpretability aspects. Neurocomputing **135**, 203–217 (2014)
4. Khajehzadeh, M., Taha, M.R., El-Shafie, A., Eslami, M.: A survey on meta-heuristic global optimization algorithms. Res. J. Sci. Eng. Technol. **3**(6), 569–578 (2011)
5. Li, W.: Design of PID controller based on an expert system. Int. J. Comput. Consum. Control (IJ3C) **3**(1), 31–40 (2014)
6. Łapa, K., Przybył, A., Cpałka, K. A new approach to designing interpretable models of dynamic systems. Lecture Notes in Artificial Intelligence, vol. 7895, pp. 523–534. Springer, Berlin (2013)
7. Łapa, K., Zalasiński, M., Cpałka, K.: A new method for designing and complexity reduction of neuro-fuzzy systems for nonlinear modelling. Lecture Notes in Artificial Intelligence, vol. 7894, pp. 329–344. Springer, Berlin (2013)
8. Malhotra, R., Sodh, R.:. Boiler flow control using PID and fuzzy logic controller. IJCSET **1** (6), 315–331 (2011)
9. Perng, J.-W., Chen, G.-Y., Hsieh, S.-C.: Optimal PID controller design based on PSO-RBFNN for wind turbine systems. Energies **7**, 191–209 (2014)
10. Rutkowski, L. Computational Intelligence. Springer, Heidelberg (2008)
11. Simon, D.: Biogeography-based optimization. IEEE Trans. Evol. Comput. **12**, 702–713 (2008)
12. Szczypta, J., Łapa, K., Shao, Z. Aspects of the selection of the structure and parameters of controllers using selected population based algorithms. In: Artificial Intelligence and Soft Computing. Lecture Notes in Computer Science, vol. 8467, pp. 440–454 (2014)

13. Tan, Y., Shi, Y., Tan, K.C. (Eds.): Fireworks Algorithm for Optimization, ICSI 2010, Part I, LNCS 6145, pp. 355–364 (2010)
14. Zalasiński, M., Łapa, K., Cpałka, K. New algorithm for evolutionary selection of the dynamic signature global features. Lecture Notes in Artificial Intelligence, vol. 7895, pp. 113–121. Springer, (2013)