Cascade PID Controller Optimization using Bison Algorithm

Anezka Kazikova[1][0000−0001−5053−843X], Krystian Łapa[2][0000−0002−3926−5685], Michal Pluhacek[1][0000−0002−3692−2838], and Roman Senkerik[1][0000−0002−5839−4263]

[1] Department of Informatics and Artificial Intelligence, Tomas Bata University, Zlin, Czech Republic
kazikova@utb.cz
[2] Department of Computational Intelligence, Czestochowa University of Technology, Czestochowa, Poland
krystian.lapa@pcz.pl

Abstract. Meta-heuristic algorithms are reliable tools for modern optimization. Yet their amount is so immense that it is hard to pick just one to solve a specific problem. Therefore many researchers hold on known, approved algorithms. But is it always beneficial? In this paper, we use the meta-heuristics for the design of cascade PID controllers and compare the performance of the newly developed Bison Algorithm with well-known algorithms like the Differential Evolution, the Genetics Algorithm, the Particle Swarm Optimization, and the Cuckoo Search. Also, in the proposed approach, the controller parameters were encoded to increase the chance of reducing the controller structure, and thus facilitate the automatic selection of its configuration. The simulations were performed for three different control problems and checked whether the use of cascade structures could bring significant benefits in comparison to the use of classic PID controllers.

Keywords: PID optimization · cascade PID controllers · meta-heuristics.

1 Introduction

The goal of a typical control system is to achieve the set state of a specific element of the controlled object. Such an element can be, e.g., the speed of a motor or the position of some mass. The difference between the current and the set state is called an offset. The control system should efficiently decrease the offset while also taking into account various other control criteria [1]. This is achieved by properly affecting the object by basing it on the control signal. The most commonly used in practice PID controllers [2] generate a control signal by amplifying the offset signal and its integral and derivative. To ensure efficient operation of the PID controller, three gain factors should be selected (proportional, integral, and differentiating). Therefore, the problem of optimizing three parameters arises, which is unfortunately often dealt with by a trial and error method, or the parameters are selected based on expert knowledge.

Control systems are constantly evolving, and new solutions in this field are emerging. Starting from FOPID controllers [3], through cascade PID controllers [4], and ending with controllers based on artificial intelligence (e.g. fuzzy systems [5], neural networks [6]) or hybrid solutions [7]. Along with their complexity and the increased number of parameters that need to be optimized, their control capabilities also increase. The cascade or fuzzy-based controllers mentioned above can already be based on a larger number of signals (not only offset), but they also allow to eliminate control symmetry and other defects appearing in standard PID controllers [8]. However, a larger number of signals means that an expert often selects the controller structure a priori.

The development of control systems increased the computational complexity and the difficulty of optimizing their parameters. Population-based meta-heuristic algorithms (PBA) seem to be suitable methods for such optimization. Their idea is based on processing a population of individuals, each representing one solution of the addressed problem, similarly as it would be dealt with in the real world. The gain of meta-heuristics lies in an accessible and quick solution, improving in the process: beneficially for ever-changing, complex systems.

Meta-heuristics usually root in mimicking bio-inspired phenomena: how did nature manage the optimization. Simulations of various principles from genetics, the theory of evolution, or movement patterns of a variety of animal species, created a significant number of optimization methods like the Differential Evolution [9], the Genetic Algorithm [10], the Particle Swarm Optimization [11], the Grey Wolf Optimizer [12], or the Cuckoo Search [13]. And since the source of inspiration is unlimited, the number of bio-inspired algorithms rises as well [14].

The continuous development of population-based algorithms makes it difficult to select one to solve a specific problem. Moreover, according to the theory of No Free Lunch in optimization, there is no single optimal method to solve all problems [15]. As a result, in many new papers, the old, well-known, and proven population algorithms such as GA, DE, PSO, and CS are being used (or their modifications) [16, 17]. However, the increase in complexity of the solved problems causes new population algorithms to find their place (see, e.g., [14, 18]). New algorithms are using a variety of different mechanisms to overcome the optimization problems [19], such as population division into subgroups with different behaviour [20], dynamic adaptation of parameters [21, 22], population restart [23], or boosting the exploration of the feasible solution area [24].

In this paper, the idea of a cascade PID controller with a dynamic structure is proposed. This approach frees the user from the need to design the exact controller structure. To cope with that, a Bison Algorithm (BA [24]) is used to optimize the controller parameters. The idea of this paper is also to check whether a new algorithm, such as BA, allows one to get better results compared to classic population-based algorithms. The results were verified on three control problems.

The structure of the paper is as follows: in Section 2 a proposed method is described, in Section 3 the simulations are described and summarized, and in Section 4 the conclusions are drawn.

2 Methods

This section describes the proposed controller structures, the method for their evaluation, and finally, the Bison Algorithm, which was used to optimize them.

2.1 Proposed controller structures

The output for typical parallel form of PID controller is calculated as follows:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(t)\,dt + K_d \cdot \frac{de(t)}{dt}, \tag{1}$$

where $u(t)$ is control signal, $e(t)$ is offset and K_p, K_i, K_d are proportional, integral and derivative gains. If the gain value is equal to zero, corresponding element of controller can be treated as reduced, which as a result allows to obtain the P, I, D, PI, PD or ID controllers. In the case of population-based algorithms, parameter values usually take random real values during initialization, so the chance that a given controller element will be initialized as a reduced one is virtually none. That is why, in this paper we propose an approach that will increase the chance of creating a population of controller solutions in which some elements are reduced at the very beginning:

$$u(t) = r(K_p) \cdot e(t) + r(K_i) \cdot \int_0^t e(t)\,dt + r(K_d) \cdot \frac{de(t)}{dt}, \tag{2}$$

where $r(K)$ is defined as follows:

$$r(K) = \begin{cases} 0 & \text{for} \quad K < K^- + \alpha \cdot (K^+ - K^-) \\ \frac{(K - K^- - \alpha \cdot (K^+ - K^-))}{(K^+ - K^-)} & \text{for} \quad \text{else} \end{cases}, \tag{3}$$

where K^- and K^+ is minimum and maximum value that K can take, α stands for coefficient setting below which value K is treated as zero. It is worth noting that the equation (3) scales the parameter K accordingly so that the final value of it will be within the appropriate range.

Cascade PID controllers use multiple PID blocks connected in cascades. The structure of such controllers depends on the given simulation problem and the number of signals that can be used. In connection with the above, further examples of problems and structures proposed for them will be described further below. The proposed structures use PID blocks by the formula (2). This approach to dynamic structure selection of cascade PID controllers is new and does not require the use of additional binary parameters or hybrid algorithms, such as in paper [1].

Water Tank Test (WTT). In this problem the purpose of the controller is to maintain the desired water level h^* in the tank by changing the water inflow. The tank has: a surface area A, a controllable water inflow q_{in}, an external water

inflow q_{ex}, an additional emergency water outflow q_{em} (e.g. when more water is needed in emergency situations) and s a water outflow q_{out}. The water level in the tank is determined as follows (see e.g. [25]):

$$\dot{h} = \frac{1}{A}\left(q_{in} + q_{ex} - q_{em} - s \cdot \sqrt{2gh}\right), \tag{4}$$

where $g = 9.81 m/s^2$ is the gravitational acceleration. The proposed controller structure for this problem is shown in Fig.1.a).

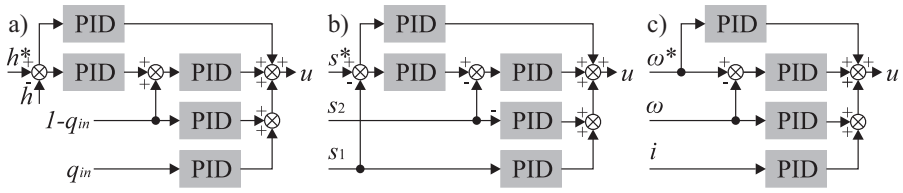

Fig. 1. Proposed cascade PID structures for control problems under consideration: a) WTT, b) MSD, C) DCM.

Mass Spring Damper (MSD). In this problem, the purpose of the controller is to maintain the desired position s^* of mass m_1 by managing the control force F. The mass is connected via spring to the mass m_2 and then by another spring to constant point y. The positions of masses are marked analogously as s_1 and s_2 and the stiffness constant k for both masses were assumed to be the same. The equations of such model are described as follows (see, e.g., [26]):

$$\begin{cases} s_1 = v_1 \cdot t + \frac{1}{2} a_1 \cdot t^2 \; v_1 = a_1 \cdot t \; a_1 = \frac{1}{m_1}\left(k \cdot (s_2 - s_1) - v_1 \cdot y\right) \\ s_2 = v_2 \cdot t + \frac{1}{2} a_2 \cdot t^2 \; v_2 = a_2 \cdot t \; a_2 = \frac{1}{m_2}\left(k \cdot (F - s_2) - v_2 \cdot y\right) \end{cases}. \tag{5}$$

The proposed controller structure for this problem is shown on Fig.1.b).

DC Motor (DCM). In this problem the purpose of the controller is to maintain the desired motor speed ω^* by managing input voltage F. The motor has: the speed ω, moment of inertia of the rotor J, viscous friction constant b, motor torque $T = K_t \cdot i$ (where i is armature current and K_t is motor torque constant), electric inductance L, electric resistance R and counter-electromotive force $e = K_e \cdot \dot{\omega}$ (where K_e is electromotive force constant). The equations of such model are described as follows (see e.g. [27]):

$$\begin{cases} \dot{\omega} = \frac{1}{J}\left(K_t \cdot i - b \cdot \omega\right) \\ \dot{i} = \frac{1}{L}\left(-R \cdot i + V - K_e \cdot \omega\right) \end{cases}. \tag{6}$$

The proposed controller structure for this problem is shown on Fig.1.c).

It should be noted that the proposed approach with the dynamic reduction of the structure will allow the reduction of entire blocks in cascade structures. Because of that, a simplified controller can be obtained, and the redundant controller's input signals will not be used. Therefore, a similar approach can be used for developing structures for other simulation problems, and the optimization algorithm should itself correct the structure and select signals that bring the most benefits in the control process.

2.2 Fitness function

To evaluate the controller, four control criteria were used, and then they were aggregated into the single-objective function, which is described in this section.

Error. The first criterion counts the sum of the offsets in time. For the first of the problem it can be written as follows:

$$error = \frac{1}{J} \sum_{j=1}^{J} |h^* (t_j) - h (t_j)|, \tag{7}$$

where t_j stands for discrete-time point ($j = 1, ..., J$), J stands for time sample. For the rest of the simulation problems, it can be defined analogously.

Overshoot. The second criterion is an overshoot, i.e. exceeding the desired water level for WTT problem (for the rest of simulation problems it can be calculated analogously):

$$over = \max_{j=1,...,J} \{h (t_j) - h^* (t_j)\}. \tag{8}$$

Oscillations. The third criterion is occurrence of oscillations calculated as the sum of changes in controller output $u (t_k)$:

$$oscs = \frac{1}{J-1} \sum_{j=2}^{J} |u (t_j) - u (t_{j-1})|. \tag{9}$$

Suit. The last criterion's purpose is to check matching the signal to the desired one. In control systems, a steady offset error may appear (an error that persists in steady-state [28]). Checking if the signal is close to the given one can eliminate such cases, and this can be calculated as follows for the WTT problem:

$$suit = \frac{1}{K} \sum_{k=1}^{K} \begin{cases} 0 \text{ for } |h^* (t_k) - h (t_k)| < \beta \\ 1 \text{ for } \qquad \text{else} \end{cases}, \tag{10}$$

where β is an acceptable offset of the signal. For the rest of the simulation problems, this criterion can be defined analogously.

Fitness. In this paper the four presented above criteria are aggregated into single-objective function, which should be minimized and it is defined as follows:

$$fitness = error \cdot w_e + over \cdot w_v + oscs \cdot w_o + suit \cdot w_s, \tag{11}$$

where **w** stands for weights of components that might differ for each simulation problem. It is worth noting that the above function does not take into account the complexity of the controller structure. Thanks to this, the optimization of parameters will not strive to obtain the simplest structure but to obtain the structure best suited for the given problem. If one wants to achieve simpler structures, one should include an additional criterion to assess the complexity of the structure.

2.3 Bison Algorithm description

The Bison Algorithm is a recent swarm algorithm inspired by the behavior of bison herds [24]. When bison are in danger, they form a circle with the strongest on the outline, trying to protect the weak ones inside. The algorithm simulates this movement by shifting the individuals closer to the center of several fittest solutions. Since bison are also persistent and remarkable runners, the algorithm devotes a small set of solutions to explore the search space; dividing the population into two groups: the swarming and the running group.

The main loop of the algorithm starts by computing the target of the swarming movement, and the swarming group moves in a direction towards the target, if it improves their quality. The running group shifts in the run direction vector, which is slightly altered after each iteration. If a runner comes upon a promising solution, better than at least one of the swarming ones, the newly discovered solution is copied to the swarming group, and it becomes the target of the next movement; otherwise, the target is computed from several fittest solutions.

Algorithm 1. Bison Algorithm Pseudocode

1. generate the swarming group randomly
   generate the running group around the best bison
   generate the $run\ direction\ =\ random(\frac{ub-lb}{45}, \frac{ub-lb}{15})_{dim}$
2. for every migration round m do
3. determine the swarming target:
4. if $f(runner_{(m-1)}) < f(swarmer_{(m-1)})$ then
5. $center = runner_{(m-1)}$
6. else
7. compute the center of the strongest solutions:
   $weight\ =\ (10, 20, 30, ..., 10 \cdot s)$
   $center\ =\ \sum_{i=1}^{s} \frac{weight_i\ \cdot\ x_i}{\sum_{j=1}^{s} weight_j}$
8. for every bison in the swarming group do
9. compute new position candidate x_{new}:
   $x_{new}\ =\ x_{old}\ +\ (center\ -\ x_{old})\ \cdot\ random(0,\ overstep)_{dim}$
10. if $f(x_{new})\ <\ f(x_{old})$ then move to the x_{new}

```
11.     end for
12.     adjust the run direction vector
        run direction = run direction · random(0.9, 1.1)_dim
13.     for every bison in the running group do
14.         move: x_new = x_old + run direction
15.     end for
16.     check boundaries
17.     if  f(x_runner) < f(x_swarmer)
18.         then copy x_runner to the swarming group
19.     sort the swarming group by f(x) values
20. end for
```

Where:

- *run direction* is the run direction vector,
- ub and lb are the upper and lower boundaries of the search space,
- x_{new} and x_{old} represent the current and the previous solutions respectively,
- and dim is a dimension,
- s is the elite group size parameter defining the number of the fittest solutions for center computation,
- *overstep* parameter defines the maximum length of the swarming movement,
- and *swarm group size* parameter sets number of bison performing the swarming movement.

3 Simulations

In the simulations, the same assumptions were made for all algorithms: number of evaluations: 25000, number of individuals in the population: 50, simulation repetitions: 100, algorithm parameters were selected following the suggestions from the literature (see e.g. [9–11, 13, 24]).

The following parameters were set: for WTT: $h^* = 1m$, $A = 4.0m^2$, $q_{out} = s \cdot \sqrt{2gh}$, $s = 0.05m^2$, simulation time $T = 100s$, time step $dt = 0.1s$, $w_e = 10$, $w_v = 0.01$, $w_o = 1.0$, $w_s = 0.1$, q_{ex} and q_{em} are shown in Fig.2.a)., for MSD: $m_1 = m_2 = 0.2kg$, $k = 10$, $y = 0.5m$, simulation time $T = 10s$, time step $dt = 0.001s$, $w_e = 1$, $w_v = 0.01$, $w_o = 0.5$, $w_s = 0.1$, s^* is shown in Fig.2.b). for DCM: $J = 0.01kg \cdot m^2/s^2$, $b = 0.1Nms$, $K_t = 0.01Nm/Amp$, $L = 0.5H$, $R = 1.0ohm$, $K_e = 0.01Nm/Amp$, simulation time $T = 8s$, time step $dt = 0.005s$, $w_e = 1$, $w_v = 0.01$, $w_oc = 0.1$, $w_s = 0.1$ and ω^* is shown in Fig.2.c).

3.1 Simulation results

Detailed simulation results are shown in Table 1, while examples of the operation of the obtained control systems are shown in Fig. 3.

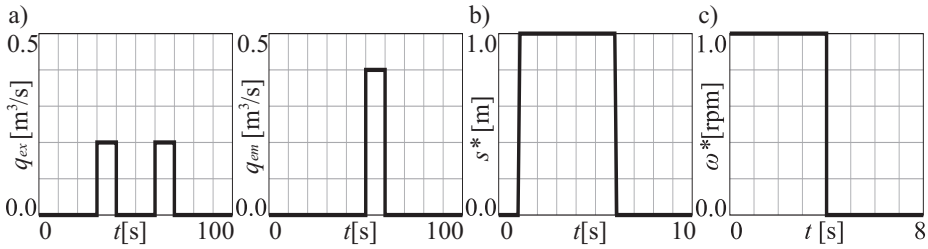

Fig. 2. Signals that vary in time for: a) WTT, b) MSD, C) DCM.

Table 1. Simulation results. Top results are marked in bold (results not worse by 5% from the best-found value in each row were considered as the top ones), and the worst results are underlined (results worse twice then the best-found value in each row were considered as the worst). Simulation time was averaged for all considered problems.

	problem	**structure**	GA	DE	PSO	CS	BA
average simulation results	WTT	PID (2)	**0.6754**	**0.6628**	0.7002	**0.6674**	**0.6679**
	WTT	Cascade PID (2)	1.0282	0.8297	1.0791	0.7917	**0.7122**
	MSD	PID (2)	0.2222	**0.1580**	0.4694	**0.1635**	**0.1608**
	MSD	Cascade PID (2)	0.2281	**0.0372**	1.8853	0.0445	0.0445
	DCM	PID (2)	**0.0245**	0.0633	**0.0246**	**0.0245**	**0.0245**
	DCM	Cascade PID (2)	0.0929	0.0249	0.1173	**0.0206**	**0.0208**
	Times in top		2	3	1	4	5
	Times in unnaceptable		2	1	3	0	0
best simulation results	WTT	PID (2)	**0.6572**	**0.6418**	**0.6414**	**0.6402**	**0.6351**
	WTT	Cascade PID (2)	0.7607	**0.6609**	**0.6576**	0.6921	**0.6488**
	MSD	PID (2)	**0.1189**	**0.1189**	**0.1189**	**0.1220**	**0.1189**
	MSD	Cascade PID (2)	**0.0348**	**0.0333**	0.0365	0.0366	**0.0341**
	DCM	PID (2)	**0.0245**	0.0633	**0.0245**	**0.0245**	**0.0245**
	DCM	Cascade PID (2)	**0.0205**	**0.0203**	**0.0205**	**0.0205**	**0.0205**
	Times in top		5	5	5	4	6
	Times in unnaceptable		0	1	0	0	0
	Average time (s)		190	193	333	328	242

3.2 Simulation conclusions

The use of cascade PID controllers has brought a significant improvement in control for MSD and DCM problems (see Table 1). Moreover, depending on the problem, optimization of the structure automatically discards redundant elements (see Fig. 3). Optimization algorithms have allowed to find good parameters at which the states of the object tend to quickly settle in the control process without the phenomenon of overshooting and other disturbances (see Fig. 3), even for the WTT problem with two external signals changing in time (see Fig. 2.a). The Bison Algorithm performed well compared to the other metaheuristics and obtained top results for most of the considered problems.

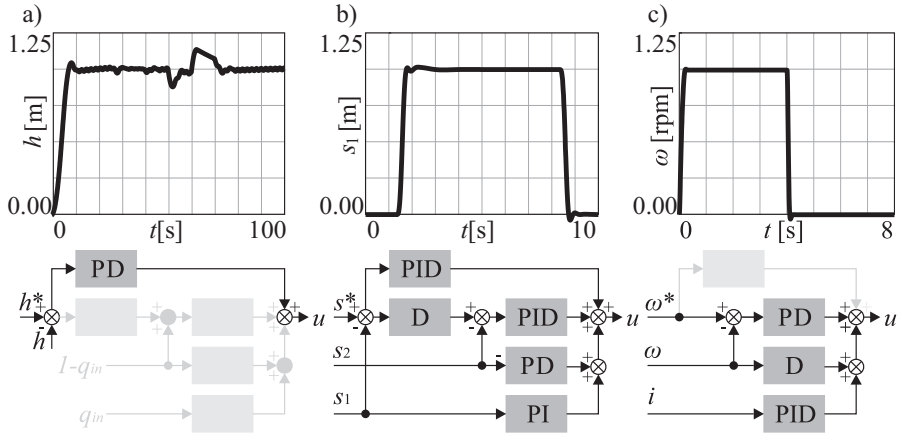

Fig. 3. Examples of the operation of the obtained control systems for: a) WTT, b) MSD, C) DCM. Reduced structure elements have been grayed out.

4 Conclusion

In this paper, we used several meta-heuristic algorithms to design PID controllers, focusing on the potential benefits of cascade layout over the classic one. We concluded that cascade PID controllers might be beneficial and that an increased number of dimensions does not harm the performance of the population-based algorithms.

On several problems of our experiment, the recent Bison Algorithm was able to outperform the Differential Evolution. The results brought us to consider the contribution of using novel meta-heuristics over the well-established and most-used optimization algorithms. Therefore, the future direction of our research should focus on more applications of metaheuristics, the novel ones included.

Acknowledgement

This work was supported by the Internal Grant Agency of Tomas Bata University under the Projects no. IGA/CebiaTech/2020/001. This work is also based upon support by COST (European Cooperation in Science & Technology) under Action CA15140, Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO). The work was further supported by resources of A.I.Lab at the Faculty of Applied Informatics, Tomas Bata University in Zlin (ailab.fai.utb.cz).

References

1. Łapa, K., Cpałka, K., Przybył, A. (2018). Genetic programming algorithm for designing of control systems. Inf. Technol. Control, 47(5), 668-683.

2. Alia, M. A., Younes, T. M., Al Subah, S. (2011). A design of a PID self-tuning controller using LabVIEW. Journal of Software Engineering and Applications, 4(03), 161.
3. Zeng, G. Q., Chen, J., Dai, Y. X., Li, L. M., Zheng, C. W., Chen, M. R. (2015). Design of fractional order PID controller for automatic regulator voltage system based on multi-objective extremal optimization. Neurocomputing, 160, 173-184.
4. Dash, P., Saikia, L. C., Sinha, N. (2015). Automatic generation control of multi area thermal system using Bat algorithm optimized PD-PID cascade controller. International Journal of Electrical Power Energy Systems, 68, 364-372.
5. Ferdaus, M. M., Anavatti, S. G., Garratt, M. A., Pratama, M. (2019). Development of c-means clustering based adaptive fuzzy controller for a flapping wing micro air vehicle. Journal of Artificial Intelligence and Soft Computing Research, 9(2), 99-109.
6. He, W., Chen, Y., Yin, Z. (2015). Adaptive neural network control of an uncertain robot with full-state constraints. IEEE transactions on cybernetics, 46(3), 620-629.
7. Łapa, K., Cpałka, K. (2017). Flexible fuzzy PID controller (FFPIDC) and a nature-inspired method for its construction. IEEE Transactions on Industrial Informatics, 14(3), 1078-1088.
8. Ang, K. H., Chong, G., Li, Y. (2005). PID control system analysis, design, and technology. IEEE transactions on control systems technology, 13(4), 559-576.
9. Back, T. (1996). Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford university press.
10. Goldberg, D.E., and Holland. J.E. 1988. Genetic algorithms and machine learning. Machine Learning 3(2). (pp. 95–99).
11. Kennedy, J. (2011). Particle Swarm Optimization. Encyclopedia of Machine Learning (pp. 760–66). Springer.
12. Mirjalili, S., Mirjalili, S.M., and Lewis, A. (2014) Grey Wolf Optimizer. Adv. Eng. Softw. 69 (March 2014), (pp. 46-61).
13. Yang, X.-S., and Deb, S. (2009). Cuckoo search via Levy flights. Proc. Of World Congress on Nature Biologically Inspired Computing (NaBIC 2009), December 2009, India. IEEE Publications, USA, (pp. 210-214).
14. Gogna, A., Tayal, A. (2013). Metaheuristics: review and application. Journal of Experimental Theoretical Artificial Intelligence, 25(4), 503-526.
15. Yang, X. S. (2012). Free lunch or no free lunch: that is not just a question? Int. J. Artificial Intelligence Tools, Vol. 21, No. 3, 1240010. DOI: 10.1142/S0218213012400106.
16. Yang, X-S., and Deb, S. (2010). Engineering optimisation by Cuckoo Search. Int. J. Mathematical Modelling and Numerical Optimisation, Vol. 1, No. 4 (pp.330–343).
17. Rahmat-Samii, Y. (2003). Genetic algorithm (GA) and particle swarm optimization (PSO) in engineering electromagnetics. In 17th International Conference on Applied Electromagnetics and Communications, 2003. ICECom 2003. (pp. 1-5). IEEE.
18. Miguel, L. F. F., Miguel, L. F. F. (2013). Assessment of modern metaheuristic algorithms–HS, ABC and FA–in shape and size optimisation of structures with different types of constraints. International Journal of Metaheuristics, 2(3), 256-293.
19. Xiong, N., Molina, D., Ortiz, M. L., Herrera, F. (2015). A walk into metaheuristics for engineering optimization: principles, methods and recent trends. international journal of computational intelligence systems, 8(4), 606-636.
20. Kadavy, T., Pluhacek, M., Viktorin, A., Senkerik, R. (2018, June). Multi-swarm optimization algorithm based on firefly and particle swarm optimization techniques.

In International Conference on Artificial Intelligence and Soft Computing (pp. 405-416). Springer, Cham.
21. Caraveo, C., Valdez, F., Castillo, O. (2017). A new meta-heuristics of optimization with dynamic adaptation of parameters using type-2 fuzzy logic for trajectory control of a mobile robot. Algorithms, 10(3), 85.
22. Ochoa, P., Castillo, O., Soria, J. (2016, September). Fuzzy differential evolution method with dynamic parameter adaptation using type-2 fuzzy logic. In 2016 IEEE 8th International Conference on Intelligent Systems (IS) (pp. 113-118). IEEE.
23. Kadavy, T., Pluhacek, M., Viktorin, A., Senkerik, R. (2017). Partial population restart of firefly algorithm using complex network analysis. In 2017 IEEE Symposium Series on Computational Intelligence (SSCI) (pp. 1-7). IEEE.
24. Kazikova, A., Pluhacek, M. Kadavy, T. and Senkerik, R. (2019) Introducing the Run Support Strategy for the Bison Algorithm. In Recent Advances in Electrical Engineering and Related Sciences: Theory and Application, edited by Ivan Zelinka. Springer.
25. Sabri, L. A., Al-mshat, H. A. (2015). Implementation of Fuzzy and PID Controller to Water Level System using LabView. International Journal of Computer Applications, 116(11).
26. Łapa, K., Szczypta, J., Venkatesan, R. (2015, June). Aspects of structure and parameters selection of control systems using selected multi-population algorithms. In International conference on artificial intelligence and soft computing (pp. 247-260). Springer, Cham.
27. Cheon, K., Kim, J., Hamadache, M., Lee, D. (2015). On replacing PID controller with deep learning controller for DC motor system. Journal of Automation and Control Engineering Vol, 3(6).
28. Rajamani, M. R., Rawlings, J. B., Qin, S. J. (2009). Achieving state estimation equivalence for misassigned disturbances in offset-free model predictive control. AIChE Journal, 55(2), 396-407.