Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 60 (2015) 1798 – 1808

Procedia Computer Science

19th International Conference on Knowledge Based and Intelligent Information and Engineering Systems

PID Tuning of Servo Motor using Bat Algorithm

Kelvinder Singh[a]*, Pandian Vasant[b], Irraivan Elamvazuthi[c], Ramani Kannan[d]

[a,c,d]*Department of Electrical and Electronnics Engineering, Universiti Teknologi PETRONAS, 31750, Malaysia*
[b]*Department of Fundamental and Applied Sciences, Universiti Teknologi PETRONAS, 31750, Malaysia*

Abstract

The Proportional-Integral-Derivative (PID) controller uses three parameters to produce the desired output of a system. The desired system performances are in terms of overshoot, rise time, settling time and steady state error. This has brought about various methods to tune the controller to the desired response. Therefore, the presence of the bat algorithm as part of the system will reduce the time and cost of tuning these parameters and improve the overall system performance.

© 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of KES International

Keywords: Metaheuristic algorithm; Proportional-Integral-Derivative (PID) controller; PID tuning algorithm; System performance; Bat Algorithm, Ziegler-Nichols; Particle Swarm Optimization

1. Introduction

A servomotor is basically a rotary actuator that is used with applications that require a motor with precise control of the angular velocity, angular position and acceleration. It is paired with a specific encoder. The Proportional-Integral-Derivative (PID) controller has been used for decades because of its simplicity and efficiency. Around 90% of the industrial controllers are built around the PID algorithm[1]. Figure 1 shows the block diagram of the PID controller with feedback also known as the closed-loop PID.

* Corresponding author. Tel.: 053687865; fax: 6053655905.
E-mail address: pandian_m@petronas.com.my

1877-0509 © 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of KES International
doi:10.1016/j.procs.2015.08.290

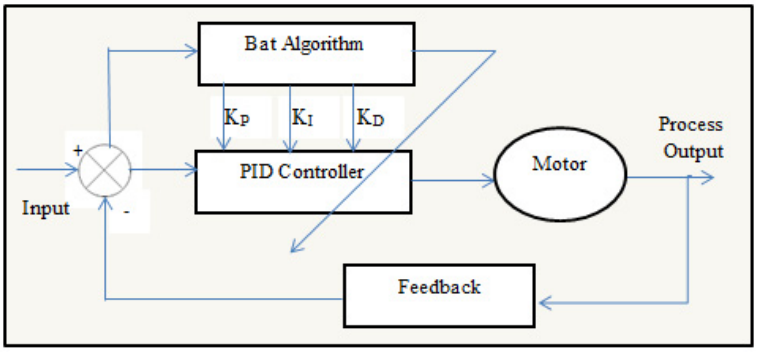

Fig. 1. Bat algorithm implementation.

Based on the Fig. 1, the PID controller is the center of the system and plays an important role in ensuring an optimum output for the system. The PID controller, also dubbed as the "three term" controller, has three main terms which consist of the proportional factor (K_P), the integral factor (K_I) and the derivative factor (K_D). These three terms will then determine the stability, steady state error, overshoot, settling time and rise time[2].

A system could produce poor performance or crash if not tuned properly[2]. It is therefore important that these parameters are properly tuned to ensure an optimum output. There are several heuristic methods that are being used to tune them. The widely used tuning methods are the Ziegler Nichols (Z-N) and Cohen Coon. These conventional PID tuning methods takes a considerable amount of time and cost. It also produces a large overshoot which is not desirable[3].Therefore new algorithms should be incorporated into the PID tuning as they are more optimized. These new algorithms are inspired from natural or man-made process. Seemingly, there is no idea that is too implausible to be turned into an algorithm[4]. Some of the algorithms are the particle swarm optimization[1,5], fruit fly algorithm[6], ant colony algorithm[7,8], cuckoo search[9,10] and the bat algorithm[11,12].

The Bat Algorithm was introduced by Yang in 2010, which was inspired by the echolocation nature of the bats. It is one of the newer additions to the "nature-inspired" algorithms. It uses the theory that a bat emits a sound pulse and then listens to it in the form of the echo bouncing back from an object whilst flying. The algorithm then uses the frequency and loudness to reach an optimum solution[7].

There is a certain benchmark function to test the effectiveness of these algorithms. One of it is known as the Rosenbrock's function and is well known as a benchmark for numerical optimization problem[13,14]. If we set the initial frequency and rate of emission to 0 and 1 respectively, while having variations of frequency, the bat algorithm becomes a standard PSO. According to Yang in[11], the Bat algorithm scored 100% while the PSO scored 98% in terms of accuracy and efficiency in the Rosenbrock's function. These functions are generally used to test the efficiency and effectiveness of the algorithm[15,16].

Other than the swarm technology, tree search algorithms have five basic algorithms which are the as naïve backtracking(BT), backjumping(BJ), conflict-directed backjumping(CBJ), backmarking(BM) and forward checking(BC)[17]. Constraint satisfaction problem(CSP), uses backtracking on the notion of tree-decomposition of the restriction networks[18] .

On the other hand, hybrid algorithms also play an important role in solving optimization problems. The hybrid differential evolution(DE)[19] has three improvements. The first modification replaces random wallk with direct exploitation local search using Hooke an Jeeves(HJ) method. The second modification replaces the harmony search with particle search optimization. The third modification is done by adding the Hooke and Jeeves method to the original cooperative hybrid. It can be seen that the second modification yielded a 15% increase in performance[20].

2. Problem statement

The PID controller is widely used in the industry and has great contributions to the control industry due to its simplicity and efficiency. It has laid a robust foundation to be applied in any plant or process. However, current conventional methods such as the Zeigler Nichols and Cohen Coon are time consuming and take up a significant amount of cost.

The new algorithm being investigated will focus on how to improve the time taken to compute the optimized values for the three parameters as well as safe cost of operating the controller. The study will also test for the feasibility of the algorithm in simulation and real application.

3. Algorithms

3.1 Particle Swarm Optimization (PSO)

PSO was inspired by having a population of candidate solutions which move around in the search space using a few pre-defined rules. The movements of these candidates are guided by their best position and the entire swarm's best known position. The movement of the swarm is then guided by improved positions. This process is a continuous one, until a reasonable solution is discovered[1]

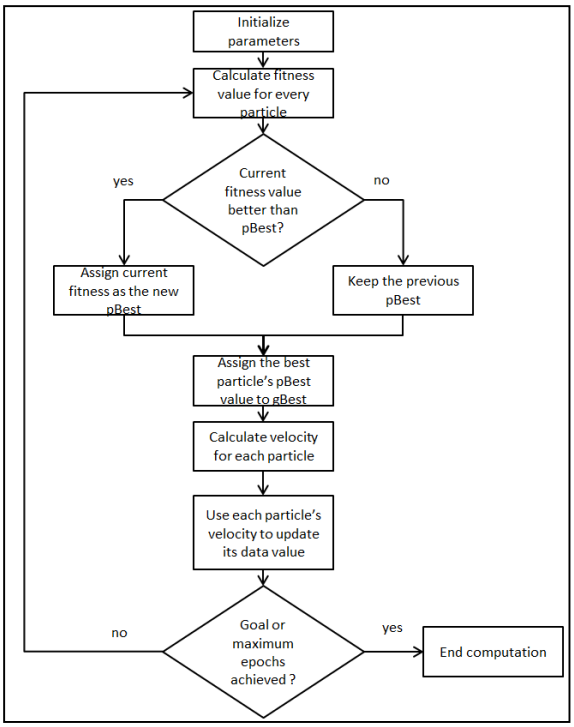

Fig. 2. Flowchart of PSO implementation.

The flowchart in Fig.2 shows the implementation of the PSO algorithm. This flowchart was then coded and used in the simulation. The parameters were initialized as shown in section 4.1.1. The population that is generated works towards optimizing the objective function.

3.2 Bat Algorithm (BA)

The BA was introduced by Yang and derived from the echolocation behavior of Bats[12]. The algorithm was simplified by three main rules:

- Bats use echolocation to approximate distance and can distinguish between prey and obstacles
- Bats fly randomly with a certain velocity at a certain position, with a constant frequency, changing wavelength and loudness to search for their prey. They can manipulate the wavelength and the pulse emission.
- The loudness has been to assumed to vary from a large (positive) number to a minimum fixed value.

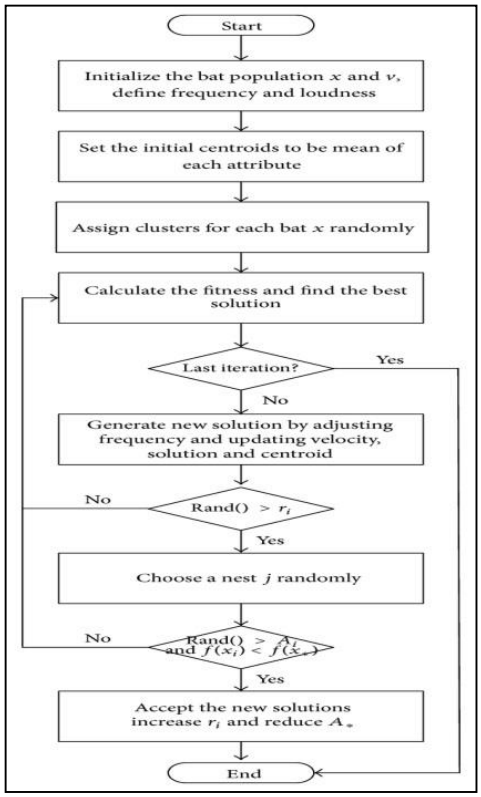

Fig. 3. Flowchart of the BA.

Fig.3 illustrates the implementation of the BA. This flowchart was then adapted into coding for simulation purposes. The parameters are initialized at the beginning of the algorithm as shown in section 4.2.1. These bats will then work together towards the objective function. In the case of this research, it will work towards optimizing the controller parameters.

4. Experimental results

4.1 Parameter Settings

The PSO and BA have parameters that need to be initialised before running the algorithm to compute the controller parameters.

4.1.1 PSO

The PSO parameters were fixed throughout the experiment. The parameter settings are shown in Fig.4. C1 and C2 are the learning factors of the algorithm.

Number of Generation: 50
Population Size: 50
C1:1.2
C2: 0.12

Fig. 4. PSO parameters.

4.2.1 BA

The BA parameters were also kept constant throughout the experiment. The parameter settings are as shown in Fig.5.

Number of Generation: 50
Population Size: 50
Amplitude: 100
Pulse: 0.5

Fig. 5. BA parameters.

4.1.3 Modelling

A servomotor model was modelled to be used in the simulation with respect to a datasheet of a real motor. This is represented by equation (3). Equation (1) and (2) are basic first and second order transfer function to examine the behavior of the algorithm.

$$H_{first}(s) = \frac{1}{s^2 + 10s + 20} \tag{1}$$

$$H_{second}(s) = \frac{1}{s^3 + 6s^2 + 5s} \tag{2}$$

$$H_{motor}(s) = \frac{0.0311}{0.00000816s^2 + 0.000144022s + 0.00096721} \tag{3}$$

Equation (1) and (2) will be used in case study one and two respectively while equation (3) will be used in the case study three which is the servomotor model case study to observe the behavior of these algorithms in tuning the controller.

4.1.4 Stopping Criteria

The stopping criteria used in this research is the Integral of Time Square Error (ITSE) shown in equation (4). It was determined that this stopping criteria provided the best system response in comparison with Integral of Absolute Error (IAE), Integral of Squared Error (ISE) and Integral of Time Absolute Error (ITAE).

$$\text{Integral of Time Square Error} = \int_0^T te(t)^2\,dt \tag{4}$$

4.2 Simulation results

Multiple simulations were carried out with different transfer functions. This was to evaluate the performance of the Bat Algorithm and Particle Swarm Optimization on different transfer functions.

4.2.1 Case Study 1: 2nd Order System

The first case study was to observe the behavior of the BA in tuning the controller for a second order system.

4.2.1.1 PSO

This section shows the response of the PSO to a step input to the system which is of second order.

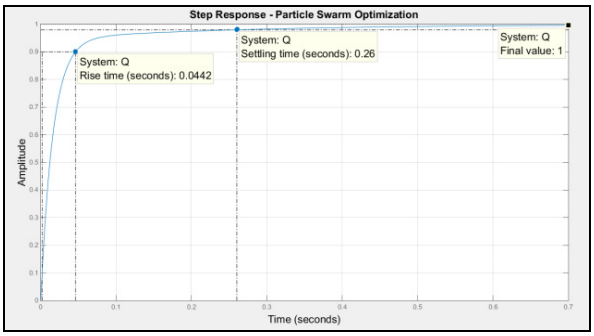

Fig. 6. step response of PSO for a second order system.

The step response shows a rise time 0.0442 seconds and a settling time of 0.26 seconds. The output of the system converges to 1.

4.2.1.2 BA

This section shows the response of the BA to a step input to the system which is of second order. The parameters were recorded for analysis.

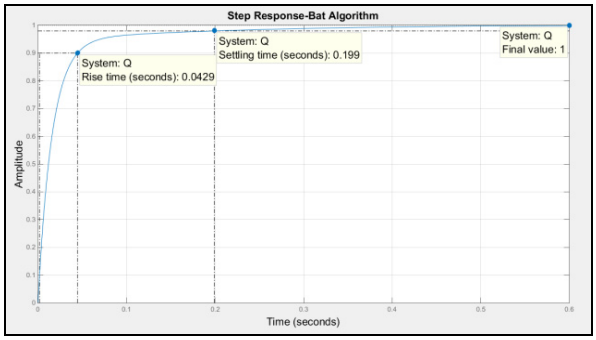

Fig. 7. Step response of BA for a second order system.

The step response shows that the BA gives a rise time of 0.0429 and a settling time of 0.199 without overshoot. The output of the system does converge to 1.

4.2.2 Case Study 2: 3rd Order System

This case study is to observe the behavior of the algorithms in tuning the controller for a third order system given a step input.

4.2.2.1 PSO

This section shows the response of the PSO to a step input applied to a system of order three.

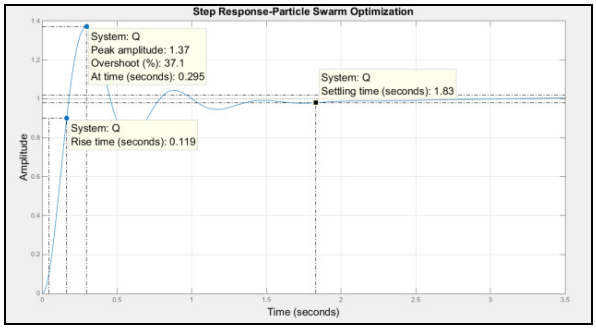

Fig. 8. Step response of PSO for a third order system.

Based on the step response, the rise time given by the PSO is 0.119 seconds and the settling time as 1.83 seconds with an overshoot of 37.1%. The system does converge to 1.

4.2.2.2 BA

This section shows the response of the BA to a step input applied to a system of order three. The parameters were recorded for analysis.

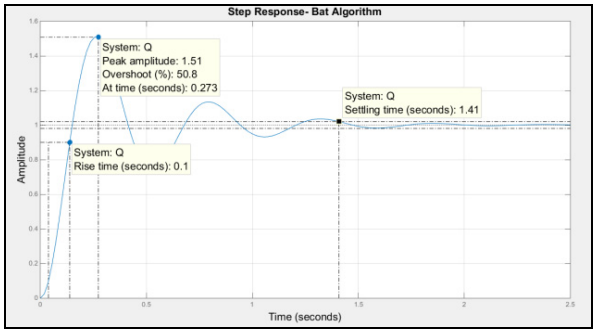

Fig. 9. Step response of BA for a third order system.

Based on the step response, the rise time given by the BA is 0.1 seconds and the settling time of 1.41 seconds with an overshoot of 50.8%. The system does converge to 1.

4.2.3 Case Study 3: Servo Motor Model

This case study is to observe the algorithm behavior for tuning the controller for a servo motor model given a step input. This servo motor transfer function is set at no load.

4.2.3.1 PSO

This section is dedicated to observe the behavior of the PSO in producing optimum controller values for a servo motor. The step response is observed and data is recorded for five runs.

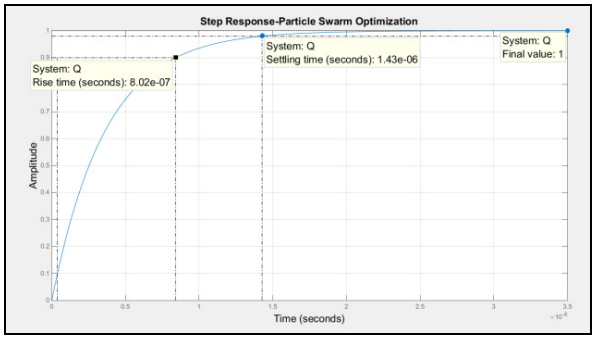

Fig. 10. Step response of PSO for the servo motor model.

Table 1. Rise time and settling time for Particle Swarm Optimization.

Run	K_P	K_I	K_D	Rise Time(s)	Settling Time(s)
1	11.2911	1.4020	900.5649	6.4e-7	1.14e-6
2	18.8388	14.9994	630.8263	9.15e-7	1.63e-6
3	59.7778	161.3995	719.4496	8.45e-7	1.43e-6
4	21.550	127.0495	672.3569	8.5e-7	1.53e-6
5	52.1229	10.4161	615.5391	9.37e-7	1.6e-6

Based on the table, five runs were conducted and the average rise time and settling was obtained. This particular algorithm had an average rise time of 8.374e-7 seconds and a settling time of 1.466e-6 seconds with no overshoot.

4.2.3.2 BA

This section is dedicated to observe the behavior of the BA in producing optimum controller values for a servo motor. The step response is observed and data is recorded for five runs.

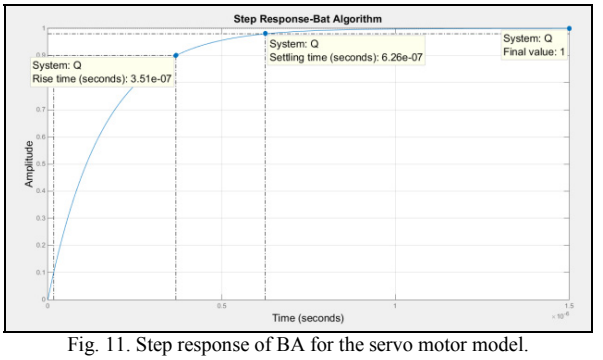

Fig. 11. Step response of BA for the servo motor model.

Table 2. Rise time and settling time for BA.

Run	K_P	K_I	K_D	Rise Time(s)	Settling Time(s)
1	5.4465	42.683	829.1195	6.95e-7	1.24e-6
2	486.23	245.32	1640.33	3.51e-7	6.26e-7
3	189.89	72.43	2896.54	1.99e-7	3.54e-7
4	243.43	74.39	2549.34	2.26e-7	4.03e-7
5	5.7329	39.93	985.33	5.85e-7	1.04e-6

The same procedure was conducted for the BA as well. The average rise time is 4.112e-7 seconds and the average settling time is 7.326e-7 seconds with no overshoot.

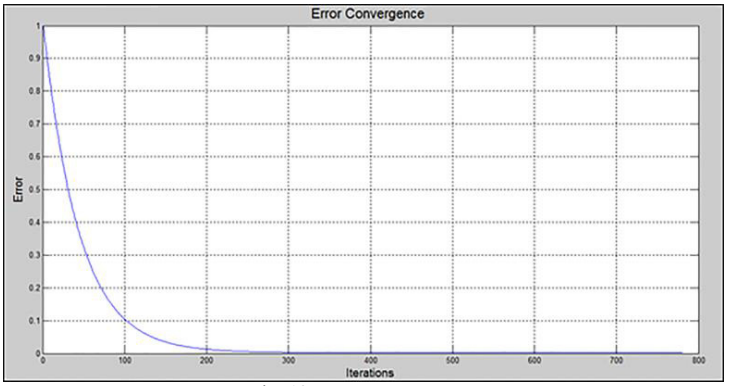

Fig. 12. Error convergence.

Fig.12 shows that the controller is performing well as the error converges to zero. This is also observed in the step response whereby the output does converge to 1.

4.3 Comparative Analysis

The results obtained in terms of rise time, settling time and overshoot were recorded in Table 3, Table 4 and Table 5. The difference in performance were calculated and analyzed.

Table 3. Rise time and settling time from case study one

Algorithm	Rise Time(s)	Settling Time(s)
PSO	0.0442	0.2600
BA	0.0429	0.1990

Table 4. Rise time, settling time and overshoot from case study two

Algorithm	Rise Time(s)	Settling Time(s)	Overshoot(%)
PSO	0.119	1.830	37.1
BA	0.100	0.141	50.8

Table 5. Rise time and settling time from case study three

Algorithm	Rise Time(s)	Settling Time(s)
PSO	8.02e-7	1.43e-6
BA	3.51e-7	6.26e-7

5. Conclusion

Based on the research conducted, it is important that a PID controller be tuned properly to ensure that a given system is working at an optimum level. It is already been discovered that previous heuristic algorithms to tune the PID controller are costly and takes up time. The "Nature-Inspired" Bat Algorithm developed by Xin-She Yang is a comparatively superior algorithm to tackle and solve complex optimization problems.

Therefore it is feasible to be considered as an algorithm for the tuning of a PID controller to control the servo motor. On a whole, it can be seen that the overall performance of the BA is greater than the PSO which produced a better rise time and settling time. However, this is not the single best solution. Future studies can focus on the motor on different load conditions as well as hybrid algorithms which can produce far more superior results as compared to a stand-alone algorithm.

Acknowledgments

The authors would like to thank Universiti Teknologi PETRONAS for sponsoring this research work.

References

1. Dashti, M., Shojaee, K., Seyedkashi, S. M. H., & Behnam, M. (n.d.). Tuning of digital PID controller using particle swarm optimization, 3383–3389. Retrieved from http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5572133
2. 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. http://doi.org/10.1109/TCST.2005.847331
3. De Almeida, G. M., e Silva, V. V. R., Nepomuceno, E. G., & Yokoyama, R. (2005). Application of Genetic Programming for Fine Tuning {PID} Controller Parameters Designed Through {Ziegler-Nichols} Technique. In *Advances in Natural Computation, First International Conference, ICNC 2005, Proceedings, Part III* (Vol. 3612, pp. 313–322). http://doi.org/doi:10.1007/11539902_37
4. Sörensen, K. (2015). Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*, *22*(1), 3–18. http://doi.org/10.1111/itor.12001
5. Suresh, A., Harish, K. V., & Radhika, N. (2015). Particle Swarm Optimization over Back Propagation Neural Network for Length of Stay Prediction. *Procedia Computer Science*, *46*(Icict 2014), 268–275. http://doi.org/10.1016/j.procs.2015.02.020
6. Han, J., Wang, P., & Yang, X. (2012). Tuning of PID controller based on Fruit Fly Optimization Algorithm. *2012 IEEE International Conference on Mechatronics and Automation*, 409–413. http://doi.org/10.1109/ICMA.2012.6282878
7. Varol, H., & Bingul, Z. (2004). A new PID tuning technique using ant algorithm. *American Control Conference, 2004. …*, *54*(I), 54–59. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1383780
8. Sekara, M., Kowalski, M.-, Byrski, A., Indurkhya, B., Kisiel-Dorohinicki, M., Samson, D., & Lenaerts, T. (2015). Multi-pheromone ant Colony Optimization for Socio-cognitive Simulation Purposes. *Procedia Computer Science*, *51*, 954–963. http://doi.org/10.1016/j.procs.2015.05.234
9. Kaveh, A., Bakhshpoori, T., & Azimi, M. (2015). Seismic optimal design of 3D steel frames using cuckoo search algorithm. *The Structural Design of Tall and Special Buildings*, *24*(3), 210–227. http://doi.org/10.1002/tal.1162

10. Wong, P. K., Wong, K. I., Vong, C. M., & Cheung, C. S. (2015). Modeling and optimization of biodiesel engine performance using kernel-based extreme learning machine and cuckoo search. *Renewable Energy*, *74*, 640–647. http://doi.org/10.1016/j.renene.2014.08.075
11. Yang, X. S. (2010). A new metaheuristic Bat-inspired Algorithm. In *Studies in Computational Intelligence* (Vol. 284, pp. 65–74). http://doi.org/10.1007/978-3-642-12538-6_6
12. Yang, X.-S. (2013). Bat Algorithm: Literature Review and Applications. *International Journal of Bio-Inspired Computation*, *5*(3), 10. http://doi.org/10.1504/IJBIC.2013.055093
13. Wang, L. W. L., Wang, X.-P. W. X.-P., & Wu, Q.-D. W. Q.-D. (2002). Ant System algorithm based Rosenbrock function optimization in multi-dimension space. *Proceedings. International Conference on Machine Learning and Cybernetics*, *2*. http://doi.org/10.1109/ICMLC.2002.1174440
14. Shang, Y.-W., & Qiu, Y.-H. (2006). A note on the extended Rosenbrock function. *Evolutionary Computation*, *14*(1), 119–126. http://doi.org/10.1162/evco.2006.14.1.119
15. Goudos, S. K., Baltzis, K. B., Antoniadis, K., Zaharis, Z. D., & Hilas, C. S. (2011). A comparative study of common and self-adaptive differential evolution strategies on numerical benchmark problems. *Procedia Computer Science*, *3*, 83–88. http://doi.org/10.1016/j.procs.2010.12.015
16. Wang, C., Duan, Q., Gong, W., Ye, A., Di, Z., & Miao, C. (2014). An evaluation of adaptive surrogate modeling based optimization with two benchmark problems. *Environmental Modelling & Software*, *60*, 167–179. http://doi.org/10.1016/j.envsoft.2014.05.026
17. Wang, X., & Xu, G. (2011). Hybrid differential evolution algorithm for traveling salesman problem. *Procedia Engineering*, *15*, 2716–2720. http://doi.org/10.1016/j.proeng.2011.08.511
18. Jégou, P., & Terrioux, C. (2003). Hybrid backtracking bounded by tree-decomposition of constraint networks. *Artificial Intelligence*, *146*(1), 43–75. http://doi.org/10.1016/S0004-3702(02)00400-9
19. Prosser, P. (1993). Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, *9*(3), 268–299. http://doi.org/10.1111/j.1467-8640.1993.tb00310.x
20. Yi, H., Duan, Q., & Liao, T. W. (2013). Three improved hybrid metaheuristic algorithms for engineering design optimization. *Applied Soft Computing Journal*, *13*(5), 2433–2444. http://doi.org/10.1016/j.asoc.2012.12.004