ISSN: 1803-3814 (Printed), 2571-3701 (Online)
https://doi.org/10.13164/mendel.2020.2.009

WHY TUNING THE CONTROL PARAMETERS OF METAHEURISTIC ALGORITHMS IS SO IMPORTANT FOR FAIR COMPARISON?

Anezka Kazikova✉, **Michal Pluhacek, Roman Senkerik**

Faculty of Applied Informatics, Tomas Bata University in Zlin, Czech Republic

kazikova@utb.cz✉, pluhacek@utb.cz, senkerik@utb.cz

Abstract

Although metaheuristic optimization has become a common practice, new bio-inspired algorithms often suffer from a priori ill reputation. One of the reasons is a common bad practice in metaheuristic proposals. It is essential to pay attention to the quality of conducted experiments, especially when comparing several algorithms among themselves. The comparisons should be fair and unbiased. This paper points to the importance of proper initial parameter configurations of the compared algorithms. We highlight the performance differences with several popular and recommended parameter configurations. Even though the parameter selection was mostly based on comprehensive tuning experiments, the algorithms' performance was surprisingly inconsistent, given various parameter settings. Based on the presented evidence, we conclude that paying attention to the metaheuristic algorithm's parameter tuning should be an integral part of the development and testing processes.

Keywords: *Parameter tuning, metaheuristics, comparison, swarm algorithms, configuration, particle swarm optimization.*

Received: 25 October 2020
Accepted: 17 November 2020
Published: 21 December 2020

1 Introduction

The comparison of algorithms is a standard method to validate new methodologies. Novel optimization methods use various performance measures to prove their efficiency. The vast majority of publications use other algorithms' results to reveal, what makes the proposed algorithm exceptional, and why to use it [9].

The problem is that many publications pay insufficient attention to the fair comparison of the compared algorithms. New promising metaheuristics are often compared to decades-old optimizers. Algorithms with carefully tuned parameters are compared to algorithms with initial parameter settings from the original proposals. Or the novel proposal compares the algorithm with data from released publications, but does not consider other parameters, like the computational capabilities of employed CPU, number of iterations, population size, or used programming language [15].

A significant number of input variables influence the results of algorithm comparisons: the dimensionality of the solved problem, population size, programming skills, CPU, number of objective function evaluations, and foremost the parameter settings of the algorithms. The final interpretation of biased, misadjusted data might have objectionable meaning.

The No Free Lunch theorem says that no optimization algorithm can find the ideal solution to all problems [16]. However, the theorem may extend even for the parameter configuration. While one set of parameters may fit the scope of one optimization task, it may be inapplicable for another [7]. Since metaheuristic algorithms use a wide range of parameters, the users often employ original parameter configurations proposed by the authors. However, no one can guarantee the general applicability of such parameters. On the contrary, many initial proposals employ only a small sample of test problems [12]. Benchmarking competitions use various recommendations for parameter settings [11]. While some prefer one set of parameters for the whole testbed [1], others require parameter tuning for each issue separately [5].

Naturally, the main focus of metaheuristic proposals is on the innovative methodologies, not on the established algorithms. Yet, despite all of this, the parameter tuning of compared algorithms is often neglected.

This paper shows that the algorithms' parameter configuration may significantly affect the final interpretation of the results. We compare four swarm algorithms in the sum of 20 parameter configurations on 30 problems of the IEEE CEC 2017 testbed and evaluate them for statistical significance.

The paper is structured as follows: Section 2 summarizes the common comparison practice. Section 3 describes the parameter tuning experiment, parameter configurations selection, the comparison on a benchmark testbed, and the results' winning configurations. Section 4 investigates the influence of the possible comparison interpretations given different parameter configurations. Finally, we conclude the findings and recommendations for future practice.

Table 1: Examined parameter configurations.

Bat Algorithm									
Source		**NP**	**Loudness**	**Pulse rate**	**Alpha**	**Gamma**	**Q_{min}**	**Q_{max}**	**Test set**
(Faris et al., 2016)	[4]	50	0.5	0.5			0	2	BAT 0
(Xue et al., 2015)	[17]	50	0.9	0.9	0.99	0.9	0	5	BAT 1
(Xue et al., 2015)	[17]	100	0.9	0.9	0.99	0.9	0	5	BAT 2
(Yang, 2010a)	[19]	50	1.5	0.5			0	2	BAT 3

Particle Swarm Optimization							
Source		**NP**	**W_{max}**	**W_{min}**	**v_{max}**[1]	**C_1=C_2**	**Test set**
(Faris et al., 2016)	[4]	50	0.9	0.2	6	2	PSO 0
(Bergh & Engelbrecht, 2006)	[2]	50	0.7298	0.7298	6[1]	1.49618	PSO 1
(Bergh & Engelbrecht, 2006)	[2]	20	0.7298	0.7298	6[1]	1.49618	PSO 2
(Harrison et al., 2017)	[6]	30	0.5	0.5	6[1]	1.9	PSO 3
(Maca & Pech, 2015)	[10]	40	0.9	0.4	95	2	PSO 4
(Bergh & Engelbrecht, 2006)	[2]	50	0.7298	0.7298	40[1]	1.49618	PSO 5
(Bergh & Engelbrecht, 2006)	[2]	20	0.7298	0.7298	40[1]	1.49618	PSO 6
(Harrison et al., 2017)	[6]	30	0.5	0.5	40[1]	1.9	PSO 7

Firefly Algorithm							
Source		**NP**	**Alpha**	**Gamma**	**Beta**	**Lambda**	**Test set**
(Faris et al., 2016)	[4]	50	0.5	1	0.2	1.0	FFA 0
(Yang, 2008)	[18]	50	0.25	1	0.2	1.0	FFA 1
(Yang, 2008)	[18]	20	0.25	1	0.2	1.0	FFA 2
(Mo et al., 2013; Yang, 2010b)	[13, 20]	50	0.2	1.5	0.2	1.0	FFA 3

Cuckoo Search					
Source		**NP**	**P_a**	**Alpha**	**Test set**
(Faris et al., 2016)	[4]	50	0.25	0.01	CS 0
(Yang & Deb, 2010)	[22]	20	0.25	1	CS 1
(Yang & Deb, 2010)	[22]	50	0.25	1	CS 2
(Faris et al., 2016)	[4]	20	0.25	0.01	CS 3

2 Common Comparison Practice

When comparing multiple algorithms, the two most frequent comparison practices distinguish the data collection procedure. While some publications apply already published results from different sources, others implement the compared algorithms themselves. Both approaches provide benefits and disadvantages.

The first approach employs data of the compared algorithms from available published results. The advantage is that these results were already approved in the review process. The researcher does not have to deal with other algorithms. Moreover, a careful choice of data source may prevent potential concerns about the algorithms' proper implementation. On the other hand, providing the same conditions as in the original experiment can get problematic. Some performance measures may be out of the question – e.g., the execution time, due to various computational sources. Moreover, since the code source implementations are often not provided, the methodology, algorithm's version, or the computing machines may be outdated.

The Passing Vehicle Search algorithm's proposal represents a beautiful example of good comparison practice based on collected published data. To provide a fair and unbiased study, the authors (Savsani and Savsani) carried out many experiments with varying parameter settings to deliver similar conditions to different scenarios [14].

The second approach employs data from the researcher's sources. They implement the algorithms themselves and can, therefore, provide all experiments in the same environment. This approach's benefits ensure equal programming skills, possible execution time measures on the same CPU, and potential employment of a universal template for all the algorithms. However, since many algorithms do not publish authentic source codes but merely pseudocodes, the implementation might be tricky. This approach also may take considerably more time. Besides the currently developed algorithm, the researcher deals with other algorithms' principles and parameter configurations. The final liability is the need for parameter tuning of unfamiliar algorithms, which is often neglected due to the time requirements. However, this paper wants to point out

[1]Since this parameter setting was originally used on unconstrained problems, the v_{max} value was not defined. However, as the IEEE CEC 2017 benchmark is bound constrained, we chose the v_{max} values as follows: v_{max}=6 as the default value from the EvoloPy library, and v_{max}=40 as 20% of the search range, as recommended by [3].

the importance of this issue and show how the parameters may affect the interpretation of the results.

3 Parameter Tuning Experiment Solution

This paper presents a showcase of the parameter tuning process on four algorithms: the Bat Algorithm (BAT) [19], the Particle Swarm Optimization (PSO) [8], the Firefly Algorithm (FFA) [20], and the Cuckoo Search Optimization (CS) [21]. The algorithms' implementations were derived from the EvoloPy optimization framework [4], and codes are available at Tomas Bata University A.I.Lab's GitHub repository[2]. The methodology was the following:

1. Find recommended parameter configurations in the literature.
2. Run the algorithms with the selected parameter settings on IEEE CEC 2017 benchmark in 10 and 30 dimensions, 51 runs each, each run consisting of 10,000·dimension evaluations of the objective function (as recommended in [1]).
3. Evaluate the performance of the parameter configurations for each algorithm separately. Check for statistical significance with the Wilcoxon rank-sum test and Friedman rank test.
4. Determine the most successful parameter settings.

3.1 Selection of Parameter Configurations

The parameter configurations were collected from various sources, including the author's parameter recommendations ([19, 18, 22]), default parameter settings in the EvoloPy optimization framework [4], adopted parameter settings [10], and parameter studies, that compared multiple configurations ([2, 6, 13, 17]). Instead of a single value, some authors recommended parameters a range of suitable parameter values (e.g., [13, 20, 22]). In such a case, our experiment employed the final configuration adopted in these publications. From extensive comparison studies (for example, [2] compares more than 25,000 parameter settings), we adopted the winning or recommended parameter combinations. Table 1 presents all the studied configurations.

Some Particle Swarm Optimization configurations initially solved unconstrained problems and left the v_{max} value undefined. Since our experiment solved continuous, bound-constrained, minimization problems, the v_{max} value was set experimentally to 6 and 40. The first one is the default EvoloPy value, and the second comes from the recommended 20% of the search space [3].

[2] https://github.com/TBU-AILab/Bison-Algorithm

3.2 Results and Statistical Interpretation

We employed two statistical tests to examine the Table 1 parameter configurations: the Friedman rank tests and the Wilcoxon rank-sum test, both with the significance level P<0.05. The Friedman rank test in Figure 1 ranks the algorithms across all the tested problems. Each algorithm over the Nemenyi Critical Distance delivered significantly worse results than the first-ranked algorithm. All the obtained results were statistically significant (Table 2). The Wilcoxon rank-sum (Table 6) test shows on how many problems from the testbed one algorithm significantly outperformed all the others. The None column counts the problems, with a similar performance of the algorithms.

Table 2: Friedman P-values (significant if P<0.05).

	BAT	PSO	FFA	CS
10 D	1.20E-13	7.39E-48	1.81E-6	8.06E-5
30 D	9.14E-27	1.17E-16	7.92E-3	7.92E-3

Bat Algorithm. The recommended parameter settings for the Bat Algorithm were consistent for both statistical tests. The BAT 3 configuration outperformed all the others.

Particle Swarm Optimization. The results of the PSO parameter tuning varied based on the dimensionality of the problem. The PSO 3 and PSO 7 configurations excelled in 10 dimensions. They both significantly outperformed all the other settings in 29 cases out of 30. A pair-wise comparison with the Wilcoxon rank-sum test (in Table 3) revealed the PSO 7 configuration's superiority.

In 30 dimensions, the statistical tests had different results. While the Friedman rank test ranked the PSO 0 the first, the Wilcoxon rank-sum test highlighted the PSO 5 configuration. Table 4 shows a pair-wise comparison between these two configurations, favoring the PSO 0 parameter setting.

Finally, Table 5 compares the winning parameter configurations. The recommendation, thus, differs based on the problem's dimensionality: PSO 7 for 10 D and PSO 0 for 30 D.

Table 3: Wilcoxon rank-sum test on pair comparison of PSO 3 and PSO 7 (α=0.05).

	PSO 3	PSO 7	None
10 D	0	0	30
30 D	6	22	2

Table 4: Wilcoxon rank-sum test on pair comparison of PSO 0 and PSO 5 (α=0.05).

	PSO 0	PSO 5	None
10 D	8	13	9
30 D	15	6	9

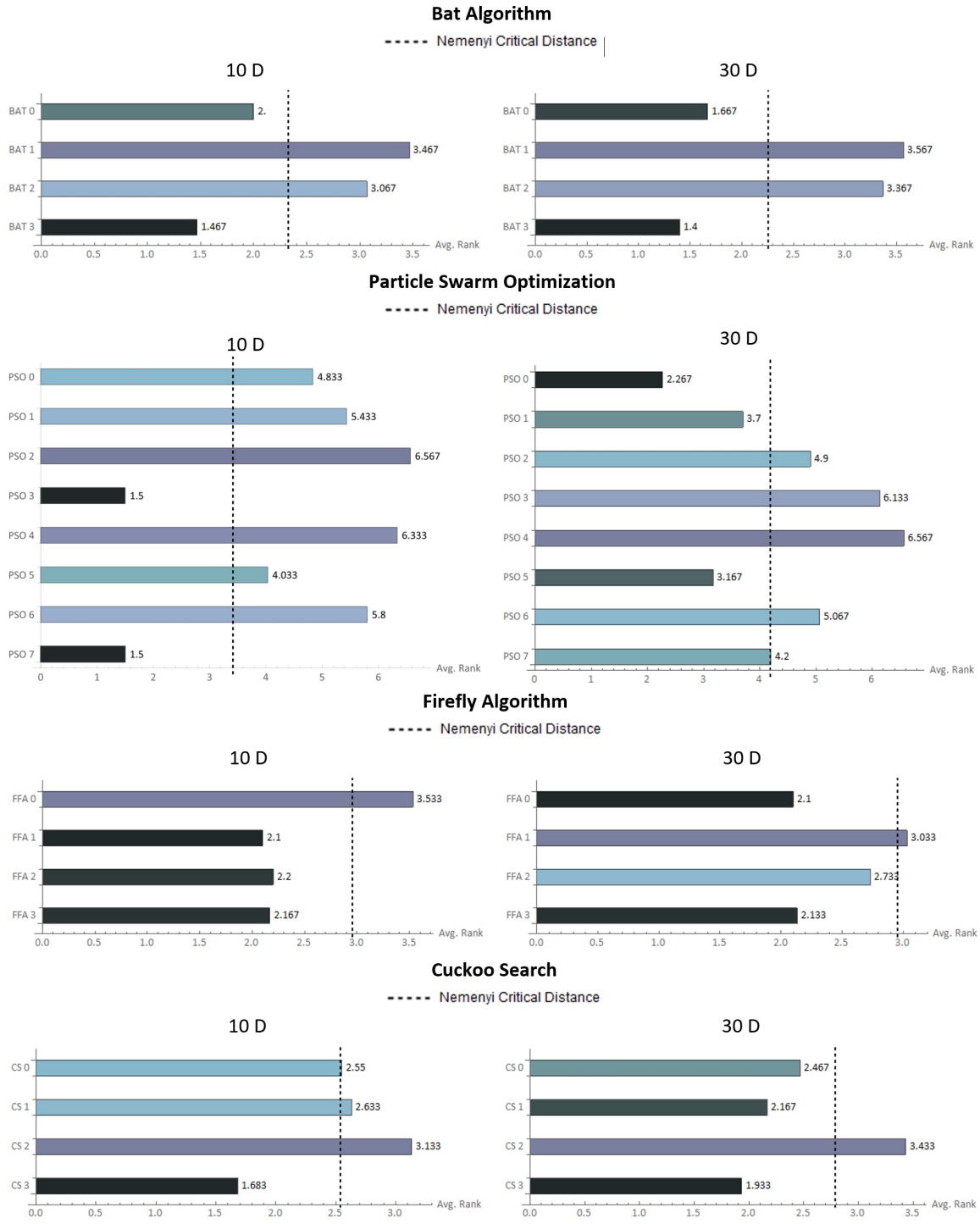

Figure 1: Friedman rank tests of the BAT, PSO, FFA, and CS algorithms with various parameter configurations.

Table 5: Wilcoxon rank-sum test on pair comparison of PSO 0, PSO 5, and PSO 7 (α=0.05).

	PSO 0	PSO 5	PSO 7	None
10 D	0	0	29	1
30 D	13	4	1	12

Firefly Algorithm. The Firefly Algorithm also had ambiguous results. While FFA 1, FFA 2, and FFA 3 delivered the best results in lower dimensions according to the Friedman rank test, in 30 D, FFA 0 significantly outperformed the other settings on 13 problems with the Wilcoxon rank-sum test. The final decision, thus, depends on the targeted dimensionality.

Cuckoo Search. The results of both statistical tests proved the superiority of the CS 3 parameter configuration.

Table 6: Results of the Wilcoxon rank-sum test comparing the parameter settings of the Bat Algorithm, Particle Swarm Optimization, Firefly Algorithm and Cuckoo Search Optimization (α=0.05) on 30 functions of IEEE CEC 2017.

Dimension	BAT 0	BAT 1	BAT 2	BAT 3	None
10 D	0	0	1	**6**	23
30 D	2	0	0	**8**	20
Sum of wins	2	0	1	**14**	43

Dimension	PSO 0	PSO 1	PSO 2	PSO 3	PSO 4	PSO 5	PSO 6	PSO 7	None
10 D	0	0	0	**29**	0	0	0	**29**	1
30 D	3	0	1	0	0	**4**	0	1	21
Sum of wins	3	0	1	29	0	4	0	**30**	22

Dimension	FFA 0	FFA 1	FFA 2	FFA 3	None
10 D	1	0	**3**	1	25
30 D	**13**	0	3	2	12
Sum of wins	**14**	0	6	3	37

Dimension	CS 0	CS 1	CS 2	CS 3	None
10 D	0	8	0	**14**	8
30 D	1	10	1	**11**	7
Sum of wins	**1**	18	1	**25**	15

Table 7: Recommended parameter configurations.

Bat Algorithm							
For dimensionality		**NP**	**Loudness**	**Pulse rate**	$\mathbf{Q_{min}}$	$\mathbf{Q_{max}}$	**Test set**
All dimensions	[16]	50	1.5	0.5	0	2	BAT 3

Particle Swarm Optimization							
For dimensionality		**NP**	$\mathbf{W_{max}}$	$\mathbf{W_{min}}$	$\mathbf{v_{max}}$[1]	$\mathbf{C_1}$=$\mathbf{C_2}$	**Test set**
30 D	[4]	50	0.9	0.2	6	2	PSO 0
10 D	[6]	30	0.5	0.5	6[1]	1.9	PSO 3 (or PSO 7)
Minimal Friedman rank sum	[6]	30	0.5	0.5	40[1]	1.9	PSO 7

Firefly Algorithm							
For dimensionality		**NP**	**Alpha**	**Gamma**	**Beta**	**Lambda**	**Test set**
30 D	[4]	50	0.5	1	0.2	1.0	FFA 0
10 D	[18]	20	0.25	1	0.2	1.0	FFA 2
Minimal Friedman rank sum	[11, 17]	50	0.2	1.5	0.2	1.0	FFA 3

Cuckoo Search					
For dimensionality		**NP**	$\mathbf{P_a}$	**Alpha**	**Test set**
All dimensions	[4]	20	0.25	0.01	CS 3

3.3 Winning Parameter Configurations

Table 7 presents the winning parameter configurations. While some parameter configurations (BAT 3, CS 3) outperformed the other settings by both statistical measures, the PSO and FFA experiments' interpretation was dependent on the solved dimension.

Hence, the winning configurations include a recommendation for 10 and 30 dimensions separately and the lowest sum of both dimensions' Friedman ranks.

The final decision for the appropriate choice of parameter configuration should reflect both the solved problems' dimensionality and the benchmark's profile. The IEEE CEC 2017 benchmarking competition examines the 10, 30, 50, and 100 dimensions [1]. In such a case, we would suggest using the configurations suited for 30 dimensions.

4 Effect of Untuned Parameter Selection

This section analyses the impact of the parameter selection. We compared three scenarios and investigated the final interpretation of the statistical results. The results were statistically significant by the Friedman rank test (Table 8) and the Wilcoxon rank-sum test with the significance level P<0.05. Yet, all the results promoted different algorithms, based on the parameter selection.

Table 8: Friedman P-values (significant if P<0.05).

	Example A	Example B	Example C
10 D	1.90E-32	1.18E-18	4.62E-17
30 D	4.73E-26	6.98E-20	2.15E-17

4.1 Example A: Parameter Selection Suited for 10 D Problems

The first scenario compared the algorithms with the parameters selected to fit the 10-dimensional problems (based on Table 7): PSO 7, BAT 3, FFA 2, and CS 3. The PSO 7 configuration reached the highest sum of wins in the Wilcoxon rank-sum test (Table 9) and could be considered the most successful algorithm. However, in 30 dimensions, the PSO 7 was significantly worse than the FFA 2 configuration (Figure 2).

Table 9: Example A: Winning algorithms on IEEE CEC 2017 with parameters suited for 10 D (Wilcoxon α=0.05).

	PSO 7	BAT 3	FFA 2	CS 3	None
10 D	**28**	0	0	0	2
30 D	1	0	**14**	11	4
Sum of wins	**29**	0	14	11	6

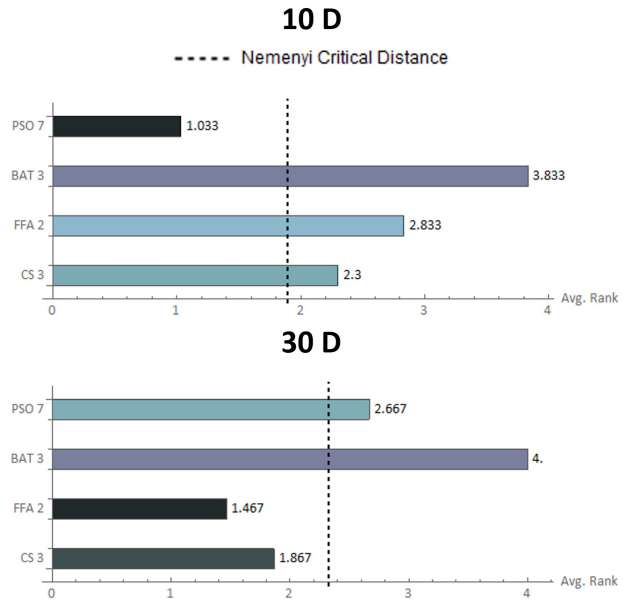

Figure 2: Example A: Friedman rank tests of the compared algorithms with parameter suited for 10 D.

4.2 Example B: Parameter Selection Suited for 30 D Problems

The second scenario investigated the algorithms' comparison with parameters selection suited for 30-dimensional problems. The parameter selection included the PSO 0, BAT 3, FFA 0, and CS 3. The Wilcoxon rank-sum test in Table 10 and Friedman rank test in Figure 3 show the superiority of the Cuckoo Search Optimization.

Table 10: Example B: Winning algorithms on IEEE CEC 2017 with parameters suited for 30 D (Wilcoxon α=0.05).

	PSO 0	BAT 3	FFA 0	CS 3	None
10 D	5	0	0	**22**	3
30 D	3	0	**13**	11	3
Sum of wins	8	0	13	**33**	6

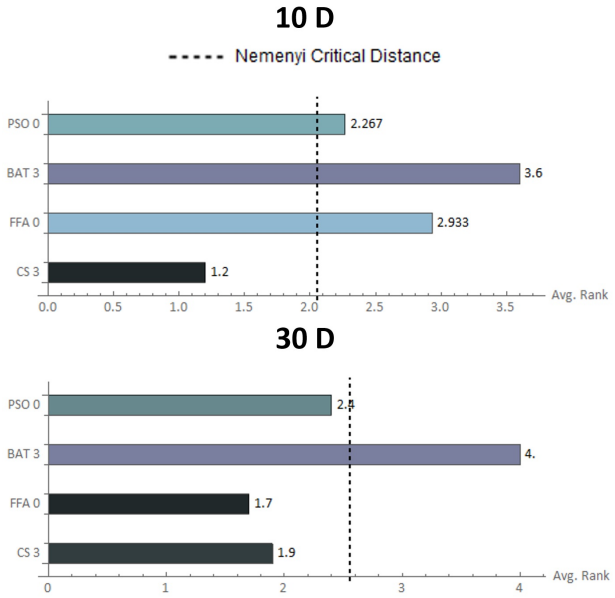

Figure 3: Example B: Friedman rank tests of the compared algorithms with parameter suited for 30 D.

4.3 Example C: Parameter Selection Based on the Sum of Friedman Ranks

The third example considers a comparison of algorithms with an inappropriate parameter selection. The experiment compared the algorithms with the worst performance measured by the maximal sum of Friedman ranks: PSO 4, BAT 1, FFA 0, CS 2. This experiment might serve as a showcase, what might happen without proper attention to the parameter setting. Once again, the statistical results in Table 11 and Figure 4 are different from previous examples: this time in favor of the Firefly Algorithm.

Table 11: Example C: Winning algorithms on IEEE CEC 2017 with a parameter selection based on the maximal sum of Friedman ranks (Wilcoxon α=0.05).

	PSO 4	BAT 1	FFA 0	CS 2	None
10 D	3	1	1	**19**	6
30 D	0	0	**25**	3	2
Sum of wins	3	1	**26**	22	8

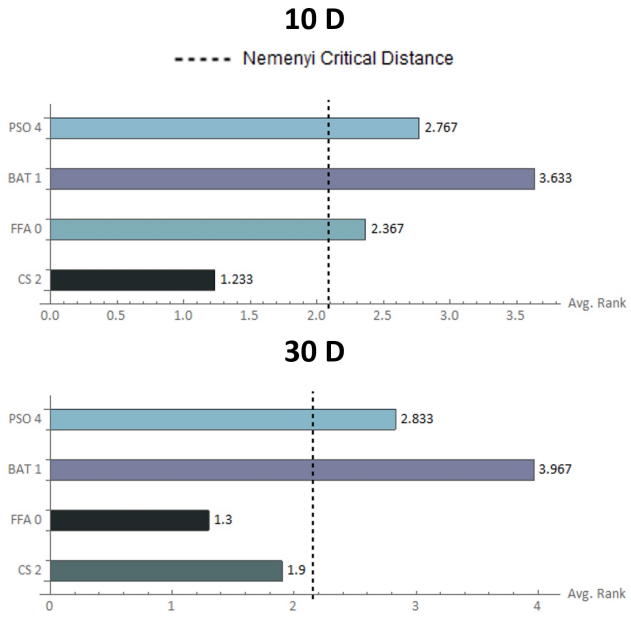

Figure 4: Example C: Friedman rank tests of the compared algorithms with inappropriate parameters (with the worst sum of Friedman ranks).

5 Conclusion

Although it is well-known that control parameter selection influences the performance of metaheuristic algorithms, the magnitude of the effect of parameter settings on the overall performance of the metaheuristic is often being misunderstood and underestimated.

This paper investigated the winning configurations of several extensive parameter tuning studies; however, the results varied substantially.

Moreover, in our experiment, three parameter selections led to three different outcomes – promoting a diverse metaheuristic every time. Our findings formed the following conclusions:

- The parameter tuning should be an integral part of the metaheuristic comparison practice.
- The choice of parameter configuration should be made with regard to the currently solved problem. While one set of parameters may fit one type of problem, it might be entirely inappropriate for another.
- Misplaced parameter configuration may produce biased results.

Since parameter tuning is a rather complicated process, the authors of novel metaheuristics should also focus on control parameter sensitivity analysis. Promoting better practice in algorithmic testing is a way to renew the metaheuristic reputation, and we hope that following these recommendations might raise the quality of testing and development processes.

Acknowledgement: This work was supported by the Internal Grant Agency of Tomas Bata University under the Projects no. IGA/CebiaTech/2020/001. 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] Awad, N., Ali, M., Suganthan, P., Liang, J., and Qu, B. Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective real-parameter numerical optimization. Technical Report, Nanyang Technological University, Singapore 34, 2016.

[2] Bergh, F., and Engelbrecht, A. A study of particle swarm optimization particle trajectories. *Information Sciences 176* (2006), 937–971.

[3] Eberhart, R., and Shi, Y. Particle swarm optimization: developments, applications and resources. In *Proceedings of the 2001 Congress on Evolutionary Computation* (2001), pp. 81–86.

[4] Faris, H., Aljarah, I., Mirjalili, S., Castillo, P., and Merelo, J. Evolopy: An open-source nature-inspired optimization framework in python. In *Proceedings of the 8th International Joint Conference on Computational Intelligence* (2016), pp. 171–177.

[5] Hansen, N., Auger, A., Finck, S., and Ros, R., 2010. Real-Parameter Black-Box Optimization Benchmarking 2010: Experimental Setup. INRIA.

[6] Harrison, K., Ombuki-Berman, B., and Engelbrecht, A. Optimal parameter regions for particle swarm optimization algorithms. In *2017 IEEE Congress on Evolutionary Computation (CEC)* (2017), pp. 349–356.

[7] Huang, C., Li, Y., and Yao, X. A survey of automatic parameter tuning methods for metaheuristics. *IEEE Transactions on Evolutionary Computation 24* (2020), 201–216.

[8] Kennedy, J., and Eberhart, R. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks* (1995), pp. 1942–1948.

[9] LaTorre, A., Molina, D., Osaba, E., Ser, J. D., and Herrera, F. Fairness in bio-inspired optimization research: A prescription of methodological guidelines for comparing meta-heuristics. *arXiv* (2020), arXiv:2004.09969.

[10] Maca, P., and Pech, P. The inertia weight updating strategies in particle swarm optimisation based on the beta distribution. *Mathematical Problems in Engineering* (2015), 1–9.

[11] Matousek, R. GAHC: Improved genetic algorithm. *Studies in Computational Intelligence 129* (2008), 507–520.

[12] Matousek, R., Popela, P., and Kudela, J. Heuristic approaches to stochastic quadratic assignment problem: Var and cvar cases. *MENDEL 23*, 1 (Jun. 2017), 73–78.

[13] Mo, Y., Ma, Y., and Zheng, Q. Optimal choice of parameters for firefly algorithm. In *2013 Fourth International Conference on Digital Manufacturing Automation* (2013), pp. 887–892.

[14] Savsani, P., and Savsani, V. Passing vehicle search (pvs): A novel metaheuristic algorithm. *Applied Mathematical Modelling 40* (2016), 3951–3978.

[15] Sorensen, K. Metaheuristics – the metaphor exposed. *International Transactions in Operational Research 22* (2015), 3–18.

[16] Wolpert, D., and Macready, W. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation 1* (1997), 67–82.

[17] Xue, F., Cai, Y., Cao, Y., Cui, Z., and Li, F. Optimal parameter settings for bat algorithm. *International Journal of Bio-Inspired Computation 7* (2015), 125–128.

[18] Yang, X.-S., Ed. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, 2008.

[19] Yang, X.-S. A new metaheuristic bat-inspired algorithm. *arXiv* (2010a), arXiv:1004.4170.

[20] Yang, X.-S. Firefly algorithm, lévy flights and global optimization. In *Research and Development in Intelligent Systems XXVI* (2010b), pp. 209–218.

[21] Yang, X.-S., and Deb, S. Cuckoo search via lévy flights. In *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)* (2009), pp. 210–214.

[22] Yang, X.-S., and Deb, S. Engineering optimisation by cuckoo search. *arXiv* (2010), arXiv:1005.2908.