End-User Challenges in Implementing Metaheuristic Algorithms from Scientific Papers

Matej Črepinšek
University of Maribor Faculty of Electrical Engineering and Computer Science
Maribor, Slovenia *matej.crepinsek@um.si*

Marjan Mernik
University of Maribor Faculty of Electrical Engineering and Computer Science
Maribor, Slovenia *marjan.mernik@um.si*

Matej Moravec
University of Maribor Faculty of Electrical Engineering and Computer Science
Maribor, Slovenia *matej.moravec@um.si*

Marko Šmid
University of Maribor Faculty of Electrical Engineering and Computer Science
Maribor, Slovenia *marko.smid2@um.si*

Miha Ravber
University of Maribor Faculty of Electrical Engineering and Computer Science
Maribor, Slovenia *miha.ravber@um.si*

Abstract

This preliminary study investigates how scientific paper components influence end-users' metaheuristic algorithms (MAs) implementation. Focusing on two papers presenting the Monkey King Evolution (MKE) and Sine Cosine Algorithm (SCA), we examined a sample of 22 Computer Science engineers tasked with implementing these MAs based on the papers and supplementary resources. Through questionnaires and statistical analysis, we assessed the impact of pseudo-code, Figures, Tables, and overall presentation on comprehension and implementation success. The findings indicate that coherent pseudo-code and visual aids, such as Figures illustrating solution movements, are critical for end-users, while inaccuracies or complex descriptions hinder implementation. The SCA paper, rated significantly higher than the MKE paper, facilitated faster and more accurate implementations. Despite access to the MATLAB source code, challenges persisted in adapting the implementations to Java, highlighting the need for platform-agnostic resources. We propose guidelines for MA papers, including detailed pseudo-code, diverse implementations, and additional implementation instructions. Although limited by a small sample, these insights justify further research into improving scientific papers for practical MA adoption.

Keywords: metaheuristics, EA, optimization, framework, implementation.

1. Introduction

The practical applicability and flexibility of MAs have driven their widespread adoption among researchers and domain experts [11], [14]. The number of new MAs has grown rapidly, with over 500 distinct MAs reported by 2023 [11]. Despite this expansion, many advanced MAs remain underutilized in practical applications, due largely to domain experts' limited knowledge, experience, and access to these methods.

There is a pressing need to accelerate the practical adoption of state-of-the-art MAs. Several factors hinder their uptake. For example, many MAs are developed in programming languages and tools, such as MATLAB, that are designed primarily for researchers and not tailored for end-users. Additionally, the vast number of algorithms creates an overchoice problem [11], [14], making it challenging to select a single MA from hundreds and identify the most suitable one for a specific problem type. Moreover, MAs are often presented with insufficient precision in scientific papers, which are often not self-contained and require field-specific knowledge, hindering successful implementation by end-users.

As researchers in the field, we receive requests frequently from end-users for MA recommendations, often accompanied by references to scientific papers describing cutting-edge methods. However, these recommendations often yield poor results, as end-users report difficulties navigating the papers. Challenges include understanding technical content, validating methods, and implementing algorithms. A key aim of this study is to identify the specific sources of these difficulties.

Scientific papers presenting novel MAs are aimed primarily at researchers advancing the field, with terminology and structure tailored to their needs rather than those of end-users. These papers typically follow a standard format: an introduction to the field and related work, a detailed description of the proposed method and its distinguishing features to enable comparison with existing approaches, experiments comparing the new method to others, and a conclusion synthesizing the findings and suggesting future work. The results are often presented through extensive Tables and graphs, supported by statistical analyses highlighting the method's advantages. However, these papers rarely include practical instructions for implementing and applying the proposed MAs.

This lack of actionable guidance poses significant challenges for end-users, who may struggle to grasp the nuances of the methods and apply them effectively. Consequently, many end-users abandon their implementation efforts, missing opportunities for innovation and improvement across various domains. This paper focuses on end-users of MAs, and investigates how specific elements of scientific papers impact their ability to understand and implement these methods.

The main contributions of this study are threefold. First, it provides empirical evidence on how paper elements, such as pseudo-code and visual aids, influence the effectiveness of end-user MA implementation, comparing the presentation in the scientific papers introducing the MKE and SCA algorithms [8, 9]. Second, we propose practical guidelines for writing MA papers, emphasizing clear descriptions, comprehensive pseudo-code, and supplementary implementations to enhance accessibility. Third, we highlight the importance of robust validation methods, demonstrating that even high-confidence implementations may differ significantly, necessitating thorough testing.

The remainder of the paper is structured as follows. Section 2 reviews related literature on MA implementation and reproducibility. Section 3 presents the empirical study, detailing the methodology and results of end-user implementation tasks. Section 4 discusses the validation of implementations using a rating-based system. Section 5 analyzes the findings, including statistical comparisons and practical implications. Finally, Section 6 summarizes the study, outlines guidelines, and suggests future research directions.

2. Related Work

The importance of effective implementation and validation of MAs based on scientific literature is well-established in the research community. A key concern is ensuring experiment reproducibility and clarity in algorithm descriptions. Researchers are increasingly encouraged to share algorithm implementations and experimental setups publicly to address these issues. Numerous studies and surveys have investigated these challenges, proposing frameworks, guidelines, and tools to facilitate accurate and reliable MA implementation.

Talbi's seminal work, *Metaheuristics: From Design to Implementation* [12], offers a comprehensive overview of MA methodologies and practical implementation guidance. It emphasizes bridging the gap between theoretical design and operational software through structured frameworks and reusable components, serving as a foundational resource for researchers and practitioners. Similarly, surveys in Computational Intelligence have examined design choices and performance trade-offs for widely used MAs, such as Genetic Algorithms (GAs), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) [10], [14]. These studies highlight critical factors for effective MA implementation across diverse applications.

Reproducibility remains a persistent challenge in MA research. Studies have criticized the frequent use of incomplete pseudo-code and the omission of critical implementation details in published papers [4], [15]. These works assess the extent to which published descriptions can be translated into functional implementations and propose methods to improve the clarity and completeness of algorithmic presentations, underscoring the need for standardized reporting practices.

To aid practitioners, several software frameworks provide standardized MA implementations, enabling rapid prototyping and testing. Notable examples include EARS, DEAP, jMetal, PyGMO, HeuristicLab, and Optuna [1], [5, 6]. These platforms support algorithm comparison and validation under consistent conditions, often integrating tools for parameter tuning, benchmarking, and visualization. Such frameworks lower the barriers to MA implementation significantly, and enhance experimental reliability.

Collectively, these efforts establish a strong foundation for rigorous and reproducible MA research. However, ambiguities in published descriptions often necessitate interpretative decisions during implementation, complicating the process. Furthermore, there is a notable lack of emphasis on the needs of MA end-users, particularly regarding accessible documentation and practical guidance tailored to their requirements.

3. An Empirical Study

This preliminary study investigates the implementation of MAs by end-users based on scientific papers. Our goal was twofold: to analyze the initial findings on how the presentation of MAs in scientific papers affects their implementation and to evaluate the methodology using a small sample. The primary research question was: Which components of a scientific paper influence the successful implementation of MAs?

3.1. Methodology

The study involved 22 Computer Science engineers enrolled in an optimization course. All the participants had experience in Java programming, had read at least one scientific paper (15 had read more than three), and were familiar with at least one MA (19 knew more than one). Additionally, all but two had previously implemented an MA. Thus, the group was relatively young, but experienced, having been introduced to MAs through education or other means.

The selection of papers was based on the following criteria:

- The paper must present a novel MA, be published in a reputable journal with a high impact factor, and be of comparable length to ensure fair evaluation.
- The selected MAs must be unknown to all the participants and have an available implementation on the MATLAB platform.
- Both papers must be published in the same year, with one having a greater impact in the literature, measured by citation counts in Web of Science and IEEE Xplore databases [3], [7].

The first two criteria ensured that the participants were unbiased by prior knowledge of the algorithms. The third criterion enabled comparison of the papers' influence in the literature.

Two papers meeting these criteria were selected, both published in 2016 in *Knowledge-Based Systems*, a high-impact journal, and of comparable length (approximately 13 pages). The first introduced the MKE algorithm [8], and the second presented the SCA [9]. Both provided MATLAB implementations. The SCA paper has a significantly higher impact, with over 45 times more citations than the MKE paper [3], [7].

Each paper was assigned randomly to 11 participants, forming the MKE and SCA groups. The participants received the questionnaire in advance to focus on specific aspects, such as reading and implementation times. The detailed study design, questions (Q1 to Q11), and question types are presented alongside their results. For each question, we calculated the mean, Standard Deviation, and median values to compare group outcomes. A statistical analysis is provided in Section 5.

This study selected two papers and 22 participants to reflect its preliminary, exploratory nature and to provide in-depth insights into end-user implementation challenges. The researchers chose the MKE and SCA papers using rigorous criteria (e.g., reputable journal, comparable length, differing citation impact) to ensure a controlled comparison. Although the small sample size limits generalizability, the findings lay a foundation for future, larger-scale research, as discussed in Section 6.

3.2. Results

The questionnaire evaluated the initial impressions, the contribution of Tables and Figures to paper comprehension, the clarity of pseudo-code, practical implementation challenges, and overall perceptions of the papers and implementation experience.

Table 1. Descriptive Statistics for Questions Q1–Q11 for the MKE and SCA Groups

	MKE	SCA	MKE	SCA	MKE	SCA	MKE	SCA	MKE	SCA	MKE	SCA
	Q1		**Q3**		**Q2**		**Q4**		**Q5**		**Q6**	
Mean	74.00	52.45	97.64	76.91	1.73	2.73	2.18	3.55	1.71	2.32	1.91	2.45
Std. Dev.	44.00	21.85	44.45	38.83	0.40	0.71	0.45	0.58	0.81	1.00	0.96	0.85
Median	46	47	80	80	2	3	2	4	1	2	2	2
	Q7		**Q8**		**Q9**		**Q10**		**Q11**		/	
Mean	2.55	2.91	131.36	90.18	60.91	36.09	2.73	3.45	51.36	85.18		
Std. Dev.	0.68	0.33	66.28	30.78	55.12	17.39	0.66	0.60	13.85	6.60		
Median	3	3	120	72	30	30	3	4	50	85		

Coherent and fluent content is typically read faster than less comprehensible material. Thus, we measured the initial reading time and time spent re-reading until the initial algorithm implementation. After each reading, participants rated paper comprehensibility from 1 to 4: 1 = Completely incomprehensible, 2 = Partially comprehensible, 3 = Mostly comprehensible, 4 = Very comprehensible.

The study questions were:

Q1: How much time did you need for the first reading of the paper (in minutes)?

Q2: Rate the comprehensibility of the paper after the first reading (scale: 1–4).

Q3: How much additional time did you spend understanding the paper after the initial implementation (in minutes)? Include time spent reviewing key sections multiple times.

Q4: Rate the comprehensibility of the paper after the initial implementation (scale: 1–4).

Table 1 for Q1 and Q3 shows that the MKE group spent an average of 171 minutes reading the paper, while the SCA group averaged 129 minutes, a difference of 42 minutes. Despite longer reading times, the MKE group rated comprehensibility as partially comprehensible, whereas the SCA group rated it as very comprehensible (Q2 and Q4 in Table 1). These results align with the assumption that coherent content is read more quickly.

Figures and Tables are critical in scientific papers, providing evidence for claims and illustrating algorithm functionality and performance. Participants rated importance from 1 to 4: 1 = Not important (did not aid understanding), 2 = Slightly important (somewhat useful), 3 = Moderately important (helpful), 4 = Very important (essential to understanding).

The questions included:

Q5: How helpful was Figure X in understanding the paper (scale: 1–4)?

Q6: How helpful was Table Y in understanding the paper (scale: 1–4)?

The MKE paper included 15 Figures and 10 tables, while the SCA paper had 15 Figures and five Tables. For the SCA paper, Figure 9 (pseudo-code mislabeled as a figure) was excluded from the analysis. Comparing the Q5 and Q6 results requires caution due to the differing number of Figures and Tables. Table 1 for Q5 and Q6 indicates that the Figures and Tables were generally rated as slightly important for comprehension. This suggests that end-users prioritized algorithm implementation over tables, which often focused on performance benchmarks. Notably, Figure 2 (MKE) and Figure 5 (SCA), both illustrating solution candidate movements in search spaces using vectors, were rated highly valuable. This highlights the importance of visual representations over purely mathematical descriptions. Similarly, Table 5 (MKE) and Tables 1 and 2 (SCA), which summarized benchmark performance, were deemed important.

Pseudo-code is a standard method for describing new algorithms, using plain language and structured formatting independent of specific programming languages. Clarity of the pseudo-code was rated from 1 to 4: 1 = Misleading, 2 = Very inadequate, 3 = Useful but inadequate, 4 = Accurate and coherent.

Q7: How would you rate the pseudo-code (scale: 1–4)?

The pseudo-code in both papers was rated as useful but inadequate (Q7 in Table 1), indicating that researchers should prioritize more detailed and coherent pseudo-code for novel MAs.

The participants implemented the algorithms using the EARS framework, a free, open-source Java-based platform for evaluating optimization algorithms [1]. They recorded implementation time and rated confidence in the correctness of their implementations. As anticipated, implementing algorithms based solely on the papers' descriptions was challenging. The participants were initially restricted to using the paper, then allowed to consult additional sources, including public repositories and direct contact with authors. All the participants obtained MATLAB source code from authors or repositories, but no Java implementations were available.

The questions included:

Q8: How much time did you spend on the algorithm's initial implementation (in minutes)?

Q9: How much additional time did you spend on the algorithm's final implementation (in minutes)?

Table 1 for Q8 and Q9 shows that the MKE group's mean implementation time for the final version was 192 minutes, while the SCA group averaged 126 minutes, a difference of 66 minutes. The implementation time reflects only net coding time, excluding breaks, web searches, and language-specific debugging. The time spent understanding the algorithm is captured in Q1 and Q3. For overall evaluation confidence in the implementations was rated from 1 to 4: 1 = Not satisfied, 2 = Partially satisfied, 3 = Satisfied, 4 = Very satisfied.

Q10: How satisfied are you with the implemented MA (scale: 1–4)?

Table 1 for Q10 indicates that both groups were generally satisfied with their implementations, with the SCA group reporting very satisfied (median: 4).

Q11: Rate the paper from 1 to 100, where 100 is the best.

The MKE group rated their paper significantly lower (mean: 51) than the SCA group (mean: 85) (Q11 in Table 1). Questionnaire comments suggested that the MKE paper's lower score stemmed from the algorithm's complexity, and imprecise descriptions, which confused the readers.

4. Validating Implementations

Typically, the goal of validating MAs is to demonstrate the superiority of one method over another. In this study, however, algorithm performance is not the primary focus. Instead, we compare different implementations of the same algorithm to assess their consistency. If the implementations are comparable, their results are expected to show no significant differences.

The stochastic nature of MAs necessitates rigorous statistical analysis, typically requiring up to 100 runs per problem to compute reliable metrics, such as mean and Standard Deviation. Validation across diverse benchmarks ensures generalizability and mitigates overfitting. This enables statistical comparisons to determine whether one implementation outperforms another significantly. However, quantifying the extent of differences can be challenging. A practical approach is to rank and rate implementations using a system inspired by competitive sports, such as tennis or chess, where the ratings reflect relative performance without necessarily indicating statistical significance. Confidence intervals, adjusted for factors like the number of tests and benchmark difficulty, enhance this evaluation.

This rating-based approach underpins CRS4EAs, a method developed for assessing algorithm performance within the EARS framework [13]. CRS4EAs employs the Glicko-2 statistical model to compute ratings.

Ratings are derived from head-to-head comparisons between implementations, treated as "players" competing to find the optimal solution for a given problem. The implementation yielding a better solution earns a win, while the other incurs a loss. A unique feature is the possibility of a draw, determined by the CRS4EAs epsilon parameter, set by default to 1e-7. If the difference in final fitness values between two implementations is less than epsilon, the result is a draw. Based on wins, losses, and draws, the system calculates expected performance and updates ratings accordingly [13].

Testing stochastic algorithm implementations poses challenges due to randomness, making it difficult to distinguish algorithmic effects from chance. Inexperienced developers often struggle with testing, especially when the initial results appear promising or surpass those from manual parameter tuning. Experienced developers first verify convergence, ensuring solutions improve over iterations. In Java, for instance, overlooking the default shallow copy behavior can overwrite the best solution in subsequent iterations.

Additional tests include comparing the implementation against a Random Search Algorithm (RS), as a proper MA should outperform random chance. Testing multiple algorithms is recommended, as performance varies by problem. For initial testing, the Sphere function is advised due to its simplicity, followed by more complex functions. Evaluating performance across diverse problems is essential, as no universal algorithm exists [15].

The benchmark comprised the CEC2014 suite with 30 continuous problems, each with a dimension of $n = 30$ and a stopping criterion of $MaxFEs = 30,000$ evaluations [1, 2].

The participants submitted implementations twice: first, based solely on the paper, and second, using additional resources, primarily the authors' MATLAB source code. The ratings and confidence intervals, defined as $\pm 2 \times RD$ (Rating Deviation), were calculated for each group. If a rating fell within the confidence interval, no statistically significant difference existed between the implementations. A rating outside the interval indicated a significant difference.

Despite access to a test run option, not all the participants submitted solutions meeting the basic requirements, such as correct class naming, exception handling, and adherence to time limits. The MKE group recorded eight successful submissions for the initial version and nine for the final version (Table 2). The SCA group recorded eight successful submissions for both versions (Table 3). Thus, three participants in each group faced difficulties in complying with the validation rules.

Table 2. Initial (Left) and Final (Right) Ratings for the MKE Implementation

	Part.	Rating	-2xRD	+2xRD
1.	SK	1935.56	1835.56	2035.56
2.	FR	1668.03	1568.03	1768.03
3.	FE	1618.28	1518.28	1718.28
4.	HU	1541.51	1441.51	1641.51
5.	GR	1528.33	1428.33	1628.33
6.	PO	1341.85	1241.85	1441.85
7.	LO	1256.19	1156.19	1356.19
8.	VO	1110.23	1010.23	1210.23

	Part.	Rating	-2xRD	+2xRD
1.	SK	1925.45	1825.45	2025.45
2.	FR	1768.34	1668.34	1868.34
3.	KO	1736.05	1636.05	1836.05
4.	HU	1522.55	1422.55	1622.55
5.	GR	1513.33	1413.33	1613.33
6.	FE	1493.08	1393.08	1593.08
7.	PO	1358.27	1258.27	1458.27
8.	VO	1306.50	1206.50	1406.50
9.	LO	1271.13	1171.13	1371.13

Table 3. Initial (Left) and Final (Right) Ratings for the SCA Implementation

	Part.	Rating	-2xRD	+2xRD
1.	PE	1924.97	1824.97	2024.97
2.	BR	1668.09	1568.09	1768.09
3.	KR	1647.62	1547.62	1747.62
4.	DR	1639.26	1539.26	1739.26
5.	SK	1616.77	1516.77	1716.77
6.	HO	1325.28	1225.28	1425.28
7.	RA	1290.11	1190.11	1390.11
8.	MO	1274.25	1174.25	1374.25

	Part.	Rating	-2xRD	+2xRD
1.	BR	1937.54	1837.54	2037.54
2.	KR	1726.35	1626.35	1826.35
3.	DR	1699.99	1599.99	1799.99
4.	MO	1681.21	1581.21	1781.21
5.	SK	1453.54	1353.54	1553.54
6.	HO	1215.34	1115.34	1315.34
7.	RA	1174.15	1074.15	1274.15
8.	PE	1111.88	1011.88	1211.88

For the MKE group, the rating range between the best and worst implementations narrowed from the initial to the final version (Table 2), suggesting greater consistency after incorporating additional resources. In contrast, the SCA group showed varied performance, with participant PE achieving the highest rating initially but the lowest in the final version (Table 3). This indicates that modifying implementations without robust validation can degrade performance inadvertently.

5. Discussion

An independent samples t-test was conducted for each question, assuming equal or unequal variances ($\alpha = 0.05$, $p > 0.05$). Significant differences were found for questions Q2 (comprehensibility after the first reading), Q4 (comprehensibility after the initial implementation), Q9 (additional time for the final implementation), Q10 (satisfaction with the implemented MA), and Q11 (overall paper rating).

The SCA group outperformed or matched the MKE group on all the questions and performed significantly better on the five noted questions. The MKE group required more time for most paper elements and reported lower overall satisfaction. This suggests that the SCA paper's clearer presentation facilitated faster comprehension and implementation, as evidenced by the shorter reading (Q1 to Q3 in Table 1) and implementation times (Q8 and Q9 in Table 1).

The participants emphasized the importance of Figures illustrating solution movements in the search space, which enhanced engagement and understanding. Only summary Tables of results were deemed useful, while detailed performance Tables were less relevant for implementation. Pseudo-code, analyzed thoroughly by all the participants, was criticized frequently for missing information. For authors of novel MAs, providing detailed and accurate pseudo-code is critical to support end-users.

Access to the source code was pivotal for successful implementation, as the participants expressed high satisfaction with their final implementations (Q10 in Table 1). However, verifying and validating MA implementations remains challenging. While MAs appear simple due to their concise algorithms, ensuring correctness is complex, influenced by factors such as random control parameters and problem type. The CRS4EAs method, designed primarily for comparing MAs, also enables validation of implementation correctness by including multiple implementation versions and benchmark MAs (EARS includes over 60 MAs) in a tournament [13]. The comparable results indicate correct implementations, while discrepancies necessitate further examination. In cases of ambiguity, competitions among implementations help assess their correctness.

Unexpectedly, significant differences persisted between implementations within the same groups, despite the high confidence levels (Q10 in Table 1). This underscores the need for thorough testing, as the initial results can be misleading. One contributing factor may be using MATLAB as the primary development platform, with pseudo-code reflecting MATLAB-specific matrix operations unfamiliar to participants accustomed to Java. This highlights the importance of platform-agnostic pseudo-code.

The EARS framework [1] played a critical role in facilitating participant implementations by providing a standardized Java-based platform for testing MAs. Its support for benchmarking and visualization, as noted in Section 2, reduced some implementation barriers, though challenges persisted due to paper-specific ambiguities (e.g., MATLAB-specific pseudo-code). Future studies could quantify the impact of such frameworks on implementation outcomes to further refine MA documentation practices.

6. Conclusion

This preliminary study explores how scientific paper components influence MA implementation by end-users. Although limited to two papers, the findings offer valuable insights and warrant further investigation. The novelty of this preliminary study lies in its empirical focus on end-user perspectives, an underexplored area in MA literature [4]. By integrating a mixed-methods approach—combining questionnaires, implementation tasks, and statistical analysis (Section 5)—we provide actionable insights into how paper elements shape implementation effectiveness, contributing to both academic research and practical MA adoption.

Pseudo-code and summary results emerged as the most critical paper components. However, the significant implementation challenges were unexpected, particularly with the MKE paper. Clear and precise algorithm descriptions, supported by detailed pseudo-code, are essential. Minor inaccuracies or omissions can lead end-users to abandon MAs, discouraging their pursuit of optimal solutions. The SCA paper was notably more coherent and better presented than the MKE paper, as reflected in the higher participant ratings (Q11 in Table 1) and confirmed by the statistical analysis. The study also addresses challenges in testing and benchmarking MAs, providing practical recommendations for implementation and paper quality.

The results confirm that well-written papers help end-users implement MAs accurately and efficiently. Based on this, we suggest keeping algorithm descriptions clear and concise, providing complete pseudo-code, offering multiple implementations (e.g., Java, Python, MATLAB) as supplementary material, using visual aids to show solution behavior, and placing non-essential content in an appendix.

Future work will expand the study to include more papers and participants. A reverse study could investigate why certain MAs gain widespread popularity, exploring factors beyond paper presentation.

Acknowledgements

The project is co-financed by the Republic of Slovenia, the Ministry of Higher Education, Science and Innovation, and the European Union - NextGenerationEU under the Recovery and

Resilience Plan, reference number C3330-22-953012 - NOO.

References

[1] EARS - evolutionary algorithms rating system (2025), https://github.com/UM-LPM/EARS, accessed: 2024-05-01

[2] Al-Roomi, A.R.: Ieee congresses on evolutionary computation repository (2015), https://www.al-roomi.org/benchmarks/cec-database

[3] Clarivate Analytics: Web of Science (2024), https://www.webofscience.com, accessed: 2024-03-13

[4] Derrac, J., García, S., Herrera, F.: Reproducibility in evolutionary computation: A study of algorithmic descriptions. Applied Soft Computing 101, pp. 107036 (2021)

[5] Durillo, J.J., Nebro, A.J.: jmetal: A java framework for multi-objective optimization. Advances in Engineering Software 42(10), pp. 760–771 (2011)

[6] Fortin, F.A., De Rainville, F.M., Gardner, M.A., Parizeau, M., Gagné, C.: DEAP: Evolutionary algorithms made easy. Journal of Machine Learning Research 13, pp. 2171–2175 (Jul 2012)

[7] IEEE: IEEE Xplore Digital Library (2024), https://ieeexplore.ieee.org, accessed: 2024-03-13

[8] Meng, Z., Pan, J.S.: Monkey king evolution: A new memetic evolutionary algorithm and its application in vehicle fuel consumption optimization. Knowledge-Based Systems 97, pp. 144–157 (2016), https://www.sciencedirect.com/science/article/pii/S0950705116000198

[9] Mirjalili, S.: Sca: A sine cosine algorithm for solving optimization problems. Knowledge-Based Systems 96, pp. 120–133 (2016), https://doi.org/10.1016/j.knosys.2015.12.022

[10] Mohapatra, P., Das, K.N.: Metaheuristic optimization algorithms for real-world electrical and civil engineering application: A review. Results in Control and Optimization 14, pp. 100349 (2024)

[11] Rajwar, K., Deep, K., Das, S.: An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges. Artificial Intelligence Review 56, pp. 13187–13257 (Nov 2023), https://doi.org/10.1007/s10462-023-10470-y

[12] Talbi, E.G.: Metaheuristics: From Design to Implementation. Wiley Publishing (2009)

[13] Veček, N., Mernik, M., Črepinšek, M.: A chess rating system for evolutionary algorithms: A new method for the comparison and ranking of evolutionary algorithms. Information Sciences 277, pp. 656 – 679 (2014)

[14] Yang, X.S.: Nature-inspired optimization algorithms: Challenges and open problems. Journal of Computational Science 46, pp. 101104 (2020)

[15] Črepinšek, M., Liu, S.H., Mernik, M.: Replication and comparison of computational experiments in applied evolutionary computing: Common pitfalls and guidelines to avoid them. Applied Soft Computing 19, pp. 161–170 (2014)