Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Adaptive Multi-paradigm Metaheuristic Optimization for Blockchain Scalability: A Comparative Evaluation of Transaction Latency and Throughput

Aminu Shawwal Adam
spsphdcsc220003@fud.edu.ng
Federal University Dutse
Muhammad Sirajo Aliyu
Federal University Dutse
Abubakar Muhammad Miyim
Federal University Dutse
Farouk Lawan Gambo
Federal University Dutse
Abdullahi Aminu Sumaila
Federal University Dutse

Research Article

Keywords: Adaptive optimization, Blockchain scalability, Latency minimization, Metaheuristic algorithm, Multi-objective optimization, Throughput maximization

Posted Date: October 30th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10869325/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License

Additional Declarations: No competing interests reported.

Abstract

Blockchain technology faces significant scalability challenges, primarily high transaction latency and limited throughput, which limiting its widespread adoption. Existing and recent optimization techniques, such as the Snake Optimization Algorithm (SOA), often lack adaptability under increasing workloads. This paper proposes the Multi-paradigm Adaptive Metaheuristic Optimization (MAMO), that integrates adaptive metaheuristic search, reinforcement-learning guidance, sparse optimization, dynamic feedback control, and game-theoretic. designed to minimize transaction latency while maximizing throughput in scalable blockchain system. An experimental comparative research design was employed using a Python-based simulation and 30 independent replications were specified for each algorithm-workload condition, with identical decision-space bounds, population size, iteration budget, and simulator conditions. across six workloads (1 to 100,000 transactions), MAMO was evaluated against SOA. Results show that MAMO consistently achieves lower mean latency (0.221% to 10.378% improvement) and higher throughput (45.041% to 94.184% improvement) across all workloads. At 100,000 transactions, MAMO achieved 0.939850 ms latency and 327,509.84 TPS, compared to 1.048680 ms and 202,864.77 TPS for SOA. Statistical analysis using the paired Wilcoxon signed-rank test with Holm correction confirms that throughput improvements are significant across all workloads ($p < 0.001$), while latency improvements are significant at workload 1 ($p = 4.696 \times 10^{-5}$). These findings provide strong evidence that MAMO offers a more effective and scalable optimization approach for blockchain systems compared to SOA.

1. Introduction

Blockchain technology has emerged as one of the most significant innovations in distributed computing by enabling secure, transparent, and decentralized transaction processing without reliance on a centralized authority [1], [2], [3]. “Blockchain system is a prime illustration of distributed ledger technologies, facilitating the recording of digital transactions in a chain of blocks” [1]. Since the introduction of Bitcoin, blockchain systems have evolved beyond digital currencies to support smart contracts, decentralized applications (dApps), supply chain management, healthcare, Internet of Things (IoT), electronic voting, and financial technologies [4], [5]. Despite these advances, blockchain scalability remains one of the principal obstacles to its widespread deployment in real-world applications requiring high transaction rates and low confirmation delays.

Scalability in blockchain is commonly evaluated using transaction latency and throughput. Transaction latency defines the elapsed time between transaction submission and confirmation, while throughput measures the number of transactions processed per second (TPS). These two-performance metrics are often conflicting because improvements in throughput frequently increase communication overhead, consensus complexity, and block propagation delay, ultimately resulting in higher latency. Consequently, achieving low latency while maintaining high throughput remains a fundamental challenge in distributed blockchain systems [6]–[8].

Recently, optimization techniques based on metaheuristic algorithms have attracted considerable attention for dynamically improving blockchain performance. Algorithms such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and the Snake Optimization Algorithm (SOA) have demonstrated promising improvements in resource allocation, consensus scheduling, and blockchain configuration [3], [18]. Nevertheless, these methods generally rely on fixed exploration–exploitation strategies and static parameter settings, limiting their ability to respond effectively to rapidly changing transaction workloads [18]. In particular, the Snake Optimization Algorithm (SOA), represents a significant advancement over classical metaheuristics by introducing a blockchain-specific, nature-inspired model rooted in the specialized mating and hunting behaviors of snakes [3]. Even though, SOA demonstrates measurable improvements in scalability by optimizing shard allocation and validation scheduling under moderate transaction loads. However, despite its domain-specific design, SOA remains a static population-based optimizer with fixed behavioral phases and without explicit control-theoretic feedback or learning mechanisms. Consequently, its adaptability diminishes as transaction volume and network scale increase, leading to latency growth under high-load conditions. These limitations reveal an important research gap. the present paper treats blockchain configuration as an adaptive optimization problem rather than as a one-time static parameter-selection problem [2], [8], [14].

The proposed Multi-paradigm Adaptive Metaheuristic Optimization (MAMO) algorithm addresses this problem by integrating population-based metaheuristic search with adaptive weighting, reinforcement learning, sparse optimization, feedback control, game-theoretic guidance, and multi-objective selection. This paper narrows to two performance outcomes: transaction latency and transaction throughput. Energy, energy efficiency, security, resource utilization, convergence, and other supporting indicators are excluded from the empirical narrative so that the paper can provide a focused assessment of the two most directly relevant scalability outcomes.

The proposed algorithm is implemented using a Python-based simulation Algorithm and evaluated against the Snake Optimization Algorithm under multiple transaction loads ranging from 1 to 100,000 transactions. Performance evaluation focuses on transaction latency and throughput, while statistical significance is assessed using the Wilcoxon signed-rank test. Experimental results demonstrate that MAMO consistently achieves lower latency and higher throughput than the baseline algorithm across all evaluated workloads. The central research question is therefore: does MAMO produce lower transaction latency and higher transaction throughput than SOA under increasing simulated transaction workloads? The contribution is fourfold. First, the paper provides a focused empirical analysis of MAMO using the two principal

scalability metrics. Second, it reports workload-dependent descriptive evidence, variability, and percentage improvements rather than relying on a single best-case observation. Third, the paper experimentally develops and evaluates an algorithm for benchmarking blockchain optimization under progressively increasing transaction workloads, demonstrating that MAMO consistently outperforms SOA in both latency reduction and throughput improvement across all evaluated transaction loads. Fourth, it validates the observed performance improvements using the Wilcoxon signed-rank test with Holm correction, providing statistical evidence of the robustness and scalability of the proposed optimization approach.

2. Related Work

Blockchain scalability research has examined architectural, consensus, and computational approaches to increasing transaction-processing capacity. Surveys identify throughput, latency, communication overhead, storage growth, and consensus efficiency as recurring scalability dimensions [2], [4], [8]. The present study focuses specifically on latency and throughput because they provide complementary measures of responsiveness and processing capacity.

Transaction latency is affected by transaction propagation, queueing, consensus execution, block generation, block propagation, and finality. Lower latency is particularly important for latency-sensitive applications where delayed confirmation directly reduces service responsiveness [5], [6]. The study's simulation therefore decomposes latency into queueing, consensus, propagation, and residual processing components before applying bounded stochastic perturbation.

Table 1
Selected literature relevant to latency and throughput optimization

Optimization Technique	Adaptive Parameter Control	Reinforcement Learning Integration	Sparse Optimization	Context Awareness	Dynamic Feedback Control	Multi-objective Optimization	Scalability Adaptation	Ref.
Genetic Algorithm (GA)	Fixed; adaptive variants available	Not integrated by default	Not supported	Not supported	Not supported	Supported through extensions	Limited	[19]
Particle Swarm Optimization (PSO)	Limited adaptive variants	Not integrated by default	Not supported	Not supported	Limited	Supported through extensions	Moderate	[20]
Ant Colony Optimization (ACO)	Primarily parameter-dependent	Not integrated by default	Not supported	Not supported	Limited	Supported through extensions	Moderate	[21]
Differential Evolution (DE)	Parameter adaptation available in variants	Not integrated by default	Not supported	Not supported	Limited	Supported through extensions	Moderate	[22]
Simulated Annealing (SA)	Temperature/cooling adaptation available in variants	Not integrated by default	Not supported	Not supported	Limited	Supported through extensions	Limited–Moderate	[23]
Whale Optimization Algorithm (WOA)	Algorithm-specific adaptive variants	Not integrated by default	Not supported	Not supported	Limited	Supported through extensions	Moderate	[24]
Grey Wolf Optimizer (GWO)	Algorithm-specific adaptive variants	Not integrated by default	Not supported	Not supported	Limited	Supported through extensions	Moderate	[25]
Harris Hawks Optimization (HHO)	Dynamic exploration–exploitation mechanism	Not integrated by default	Not supported	Not supported	Limited	Supported through extensions	Moderate	[25]
Arithmetic Optimization Algorithm (AOA)	Parameter-dependent; adaptive variants available	Not integrated by default	Not supported	Not supported	Limited	Supported through extensions	Moderate	[27]
Honey Badger Algorithm (HBA)	Algorithm-specific adaptive mechanism	Not integrated by default	Not supported	Not supported	Limited	Supported through extensions	Moderate	[28]
Snake Optimization Algorithm (SOA)	Partially adaptive	Not integrated by default	Not supported	Not supported	Limited	Single-objective by default	Good	[3]
Proposed MAMO Design	Fully adaptive	Integrated	Integrated	Integrated	Integrated	Integrated	High	This Study

Transaction throughput is commonly defined as the number of valid transactions successfully processed and committed per unit time. Throughput depends on block size, block-generation interval, consensus efficiency, validator population, transaction complexity, communication bandwidth, and hardware resources [3], [7]. Empirical blockchain evaluations demonstrate that different platform configurations can exhibit substantially different throughput and latency profiles, reinforcing the importance of controlled benchmarking [9].

Metaheuristic optimization provides an alternative to purely architectural scalability interventions by searching large and nonlinear configuration spaces. Genetic algorithms, particle swarm optimization, ant colony optimization, and other population-based methods are widely used for difficult optimization problems [10–12]. The Snake Optimization Algorithm (SOA) is a more recent nature-inspired method and has been investigated specifically for blockchain scalability, including latency and throughput improvement [13].

The limitation identified in the study is not that conventional metaheuristics are incapable of improving performance, but that many employ comparatively fixed search behavior. MAMO attempts to address this limitation by adapting search decisions according to the observed operating state. This positioning is consistent with broader research on adaptive control, reinforcement learning, and multi-objective optimization [14–17].

The research gap limitations indicate that, further improvements require optimization strategies capable of integrating adaptive multi-paradigm optimizer that jointly targets latency and throughput under systematically increasing workloads.

3. Proposed Method

3.1 Research design

The study employs a design-science and simulation-based experimental comparative design. The computational artifact is MAMO, while the blockchain simulator is the common evaluation environment. MAMO and SOA solve the same constrained configuration problem and use the same simulator, workload set, decision-variable bounds, population size, iteration budget, and reporting metrics. The intended experimental difference is therefore the optimization update mechanism rather than a change in the underlying performance model.

3.2 System Architecture

The experimental workflow, illustrated in Fig. 1, where MAMO and SOA are implemented and executed in parallel under identical simulation conditions. This design ensures that performance differences can be attributed to algorithmic enhancements rather than variations in experimental conditions. The workflow progresses from environment setup and workload definition through parallel algorithm execution, paired data collection, and culminating in statistical analysis using the paired Wilcoxon signed-rank test with Holm correction.

3.3 Simulation model

For a decision vector x and transaction workload n, the simulator is represented by the mapping $\Phi(x,n) \rightarrow (L(x,n), TPS(x,n))$, where L is transaction latency and TPS is transaction throughput. The latency model decomposes deterministic latency as:

$L(x,n) = L_queue(x,n) + L_consensus(x,n) + L_propagation(x,n) + L_processing(x,n)$.

The study models throughput as a configuration-dependent capacity modified by utilization and a stochastic factor: $TPS(x,n) = T_max(x,n) \times \eta_T(x,n) \times \xi_T$.

The stochastic layer applies bounded multiplicative perturbations to latency and throughput. Specifically, the study specifies log-normal factors with standard deviation 0.015 for both latency and throughput. This stochastic construction is intended to represent controlled experimental variability while retaining reproducibility through deterministic seed initialization.

3.4 MAMO optimization mechanism

MAMO combines adaptive metaheuristic exploration and exploitation with reinforcement-learning adjustment, sparse optimization, PID-based feedback, game-theoretic guidance, adaptive objective weighting, and Pareto-based selection. For the present paper, these components are treated as mechanisms that generate candidate blockchain configurations; latency and throughput are the two outcome variables used to evaluate those configurations.

The two principal optimization objectives are: minimize $f_1(x,n) = L(x,n)$, *maximize* $f_2(x,n) = TPS(x,n)$.

For implementation in a minimization-oriented multi-objective optimizer, throughput is represented with the negative sign: $F(x,n) = [L(x,n), -TPS(x,n)]$.

Table 2
Experimental configuration

Experimental factor	Specification
Blockchain optimization methods	MAMO; SOA
Workloads	1; 10; 100; 1,000; 10,000; 100,000 transactions
Independent replications	30 per algorithm-workload condition
Population size	25 candidate configurations
Maximum iterations	60
Base random seed	20260808
Shard bounds	2–64 (integer)
Block interval	0.5–64 ms
Validator bounds	10–200 (integer)
Block size	10–2,600 (integer)
PID parameters	Kp = 0.30; Ki = 0.025; Kd = 0.12
Simulation environment	Python / Google Colaboratory or equivalent Python environment

Thirty independent replications were specified for each of the two algorithms at each of six workload levels, giving 2 × 6 × 30 = 360 algorithm-workload observations. MAMO and SOA observations were paired by workload and replication. The study also specifies deterministic seed construction so that corresponding algorithm evaluations can share controlled stochastic streams.

3.5 Performance measures

Transaction latency is measured in milliseconds and represents the simulated time required for transaction processing. Transaction throughput is measured in transactions per second. Relative latency improvement is defined as: I_L = *100 × (L_SOA − L_MAMO) / L_SOA*.

Relative throughput improvement is defined as: I_T = *100 × (TPS_MAMO − TPS_SOA) / TPS_SOA*.

3.6 Statistical analysis

The study uses the paired Wilcoxon signed-rank test at α = 0.05 because MAMO and SOA observations are paired under corresponding workload and replication conditions. For a given metric X, the paired difference is D_r = X_MAMO,r − X_SOA,r. The Wilcoxon procedure tests whether these paired differences are systematically centered around zero.

The study further reports Holm correction for multiple comparisons and aggregate probability obtained by combining workload-specific Wilcoxon evidence.

4. Results and Discussion

4.1 Comparative Latency and Throughput Performance of MAMO and SOA

Table 4.3 presents the comparative descriptive statistics for latency and throughput obtained from MAMO and the Snake Optimization Algorithm (SOA) across the six experimental workload levels. The results indicate a consistent latency advantage for MAMO across all workload conditions.

Table 3
MAMO vs SOA Latency and Throughput Performance

Workload	MAMO Latency Mean (ms)	MAMO SD (ms)	SOA Latency Mean (ms)	SOA SD (ms)	MAMO Throughput Mean (TPS)	MAMO SD (TPS)	SOA Throughput Mean (TPS)	SOA SD (TPS)
1	0.898386	0.016827	0.918278	0.035287	321,909.54	20,681.77	165,775.45	76,115.54
10	0.897415	0.014394	0.901326	0.061483	324,448.42	16,298.05	223,693.86	80,552.44
100	0.894649	0.016849	0.898949	0.026928	328,210.92	13,550.73	203,045.12	83,746.27
1,000	0.900412	0.020471	0.905764	0.031006	327,179.01	12,571.29	185,491.18	96,090.37
10,000	0.908825	0.027974	0.910835	0.056352	328,246.20	11,114.11	211,627.64	121,014.16
100,000	0.939850	0.030920	1.048680	0.294348	327,509.84	11,836.94	202,864.77	90,799.40

For MAMO, the mean latency ranges from 0.894649 ms at a workload of 100 transactions to 0.939850 ms at 100,000 transactions. The lowest observed mean latency occurs at the workload of 100, while the highest occurs at 100,000 transactions. Although MAMO latency increases slightly as the workload becomes very large, the increase remains relatively small. The corresponding standard deviations range from 0.014394 to 0.030920 ms, indicating relatively low variability in MAMO latency across the experimental replications. In comparison, SOA records mean latency values ranging from 0.898949 ms to 1.048680 ms. At every workload level, SOA records a higher mean latency than MAMO. The difference becomes particularly evident at the highest workload, where SOA reaches 1.048680 ms, compared with 0.939850 ms for MAMO. Furthermore, SOA exhibits substantially greater variability, particularly at workloads of 10 and 100,000, with standard deviations of 0.061483 ms and 0.294348 ms, respectively.

Table 4
Comparative analysis of MAMO improvements over SOA

Workload	Latency Improvement (%)	Throughput Improvement (%)
1	2.166	94.184
10	0.434	45.041
100	0.478	61.644
1,000	0.591	76.385
10,000	0.221	55.106
100,000	10.378	61.442

figure 4.4 demonstrates that MAMO maintains lower mean latency than SOA across all six workload levels. The difference is relatively small at moderate workloads but becomes more pronounced at 100,000 transactions. At this workload, MAMO records 0.939850 ms, compared with 1.048680 ms for SOA. The graph in Fig. 4.2 therefore provides visual evidence of MAMO's ability to maintain lower transaction-processing delay as workload increases.

The throughput results in Table 4.3 show an even more pronounced difference. MAMO maintains a relatively high throughput across all workload levels, with mean values ranging from 321,909.54 TPS to 328,246.20 TPS. The highest MAMO throughput is observed at the 10,000-workload level, while the lowest occurs at workload 1. The standard deviations range from 11,114.11 to 20,681.77 TPS, indicating comparatively controlled variability.

SOA, on the other hand, produces considerably lower throughput, ranging from 165,775.45 TPS to 223,693.86 TPS. Its standard deviations are also considerably larger, reaching 121,014.16 TPS at the 10,000-workload level. Thus, descriptively, MAMO demonstrates both lower latency and substantially higher throughput, while also exhibiting more stable performance across repeated experimental observations.

These results provide preliminary descriptive evidence that the adaptive multi-paradigm search strategy employed by MAMO is more effective than SOA under the simulated blockchain operating conditions. However, whether these observed differences are statistically significant is established subsequently through the inferential statistical analysis.

Figure 4.3 provides the most pronounced visual distinction between the two algorithms. MAMO remains consistently around 322,000–328,000 TPS, whereas SOA remains substantially lower, between approximately 166,000 and 224,000 TPS. This indicates that MAMO sustains considerably greater transaction-processing capacity across all experimental workload levels.

5. Statistical evidence

The experimental design provides 30 paired observations for each algorithm workload combination. To test whether the observed differences between MAMO and SOA are systematic, we applied the paired Wilcoxon signed-rank test separately for latency and throughput at each workload. Because six workload comparisons are performed per metric, we used Holm's correction to control the family-wise error rate at α = 0.05. Effect sizes are reported as Cohen's *d < sub>z</sub>*.

5.1 Latency

Table 5. summarises the latency-specific Wilcoxon results. The mean difference (MAMO – SOA) is negative for all workloads except workload 10, indicating that MAMO tends to produce lower latency. However, statistical significance (Holm-corrected) is reached only for workload 1 ($p = 4.696 \times 10^{-5}$, *d < sub>z</sub>* = − 1.078). The remaining five workloads do not reach significance after multiplicity adjustment, indicating that the latency advantage of MAMO is largely workload-dependent.

Table 5
Paired Wilcoxon results for transaction latency (N = 30 per workload)

Workload	Mean Diff (MAMO – SOA)	Std Diff	Wilcoxon W	Raw p-value	Holm p-value	Cohen's d < sub>z</sub>	Significant (Holm)
1	−0.019892	0.018460	28	2.762×10^{-6}	4.696×10^{-5}	−1.078	Significant
10	−0.003911	0.047089	206	0.5978	1.0000	−0.0831	No Significant
100	−0.004300	0.010079	137	0.04971	0.2486	−0.4266	No Significant
1000	−0.005352	0.010535	108	0.009301	0.06511	−0.5081	No Significant
10000	−0.002010	0.028377	224	0.8712	1.0000	−0.07083	No Significant
100000	−0.108829	0.263429	133	0.04049	0.2429	−0.4131	No Significant

To assess the overall evidence for latency across all workloads, we applied Fisher's combined probability test to the six raw p-values. The resulting statistic is $\chi^2(12) = 48.675$, yielding an aggregate $p = 2.384 \times 10^{-6}$. This extremely small value indicates that, collectively, the six latency comparisons provide strong evidence of a systematic difference between MAMO and SOA, even though only workload 1 is individually significant after Holm correction.

5.2 Throughput

Table 6 shows the corresponding throughput results. The mean difference (MAMO – SOA) is positive for all workloads, indicating consistently higher throughput for MAMO. All six comparisons are statistically significant after Holm correction, with raw p-values ranging from 1.863×10^{-9} to 6.918×10^{-6}. Effect sizes are large (Cohen's *d < sub>z</sub> > >* 1.06 for all), confirming a practically meaningful advantage.

Table 6
Paired Wilcoxon results for transaction throughput (N = 30 per workload)

Workload	Mean Diff (MAMO – SOA)	Std Diff	Wilcoxon W	Raw p-value	Holm p-value	Cohen's d < sub>z</sub>	Significant (Holm)
1	156134.09	55433.77	0	1.863×10^{-9}	5.588×10^{-8}	+ 2.817	Yes
10	100754.56	64254.39	1	3.725×10^{-9}	9.686×10^{-8}	+ 1.568	Yes
100	125165.80	70195.55	4	1.304×10^{-8}	2.868×10^{-7}	+ 1.783	Yes
1000	141687.83	83519.08	5	1.863×10^{-8}	3.912×10^{-7}	+ 1.696	Yes
10000	116618.56	109902.48	34	6.918×10^{-6}	1.038×10^{-4}	+ 1.061	Yes
100000	124645.07	78978.21	5	1.863×10^{-8}	3.912×10^{-7}	+ 1.578	Yes

Fisher's combined test on the six throughputs p-values yields $\chi^2(12) = 210.287$ with an aggregate $p = 2.440 \times 10^{-38}$. This overwhelmingly strong aggregate evidence, together with the consistent significance at every workload, establishes that MAMO achieves systematically higher transaction throughput than SOA under the simulated conditions.

5.1 Limitations

While the experimental results demonstrate the effectiveness of MAMO in a simulated environment, several limitations must be acknowledged. First, the evaluation is simulation-based and therefore bounded by the assumptions, parameterization, and stochastic structure of the

blockchain simulator. The observed throughput and latency values should not be interpreted as direct measurements of a production blockchain network without independent calibration or testnet validation. Second, SOA serves as the primary baseline; additional comparisons with other contemporary metaheuristics (e.g., GA, PSO, ACO, WOA) would provide a broader context for the reported improvements. Third, the aggregate statistical significance values reported in Section 4.4 ($p = 2.384 \times 10^{-6}$ for latency and $p = 2.440 \times 10^{-38}$ for throughput) are estimated combined-probability measures derived from Fisher's method applied to six workload-specific Wilcoxon tests. Fourth, the study focuses exclusively on latency and throughput; other important scalability dimensions such as energy consumption, energy efficiency, security, and decentralization are not addressed in this paper. fifth, the Holm correction was applied within each metric (six comparisons per metric), but the study does not perform a global correction across all metrics simultaneously.

5.2 Conclusion

This paper presented the Multi-paradigm Adaptive Metaheuristic Optimization (MAMO) algorithm and evaluated its performance against the Snake Optimization Algorithm (SOA) for improving blockchain scalability. The experimental results, based on 30 independent replications across six workloads ranging from 1 to 100,000 transactions, demonstrate that MAMO consistently achieves lower transaction latency and higher throughput compared to SOA.

5.3 Key Descriptive Findings

From the comparative analysis, MAMO achieved latency improvements ranging from 0.221% at workload 10,000 to 10.378% at workload 100,000, and throughput improvements ranging from 45.041% at workload 10 to 94.184% at workload 1. At the highest workload of 100,000 transactions, MAMO achieved 0.939850 ms mean latency and 327,509.84 TPS, compared with SOA's 1.048680 ms and 202,864.77 TPS.

5.4 Statistical Evidence

The paired Wilcoxon signed-rank test with Holm correction ($\alpha = 0.05$) revealed that:

1. Throughput improvements are statistically significant across all six workloads ($p < 0.001$ for all comparisons; Holm-corrected p-values ranging from 5.588×10^{-8} to 1.038×10^{-4}), with large effect sizes (Cohen's $d < sub > z < /sub > > 1.06$).
2. Latency improvements are statistically significant only at workload 1 ($p = 4.696 \times 10^{-5}$, Cohen's $d < sub > z < /sub > = -1.078$). The remaining five latency comparisons did not reach statistical significance after Holm correction, indicating that MAMO's latency advantage is workload-dependent.
3. The Fisher combined probability test provided strong aggregate evidence for both metrics: $\chi^2(12) = 48.675$, $p = 2.384 \times 10^{-6}$ for latency, and $\chi^2(12) = 210.287$, $p = 2.440 \times 10^{-38}$ for throughput. These aggregate results confirm that, collectively across all workloads, MAMO demonstrates systematic differences from SOA in both latency and throughput.

5.5 Overall Assessment

The findings support the effectiveness of adaptive multi-paradigm optimization as a promising approach for blockchain configuration optimization. MAMO's consistent throughput advantage across all workloads, coupled with its lower latency at the most demanding workload condition, demonstrates that the integrated adaptive search mechanisms including reinforcement learning, PID feedback control, and Pareto-based multi-objective selection provide meaningful performance benefits over the fixed strategy of SOA.

5.6 Future Work Directions

Future research should extend the evaluation to additional performance dimensions such as energy consumption, energy efficiency, and security; validate the optimized configurations on private blockchain networks or public testnets; conduct ablation studies to quantify the contribution of individual MAMO components; and perform formal convergence analysis using iteration-level best-fitness histories. Additionally, comparing MAMO with a broader range of metaheuristics (GA, PSO, WOA, GWO) would provide a more comprehensive benchmarking framework.

Declarations

Conflict of Interest

i. The authors have no conflict of interest.

Ethics Approval

ii. This study does not involve human participants, animal subjects, or sensitive data.

Funding

No external funding was received for conducting this study.

Author Contribution

All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by [Aminu Shawwal Adam] and [Muhammad Sirajo Aliyu]. The first draft of the manuscript was written by [Aminu Shawwal Adam] and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Data Availability

The datasets generated and analysed during the current study, including the raw simulation observations and statistical analysis data for the MAMO and SOA algorithms, are available from the corresponding author on reasonable request.

Code Availability

The Python implementation of the MAMO algorithm and the simulation environment are available from the corresponding author on reasonable request. The implementation was designed for Google Colaboratory or an equivalent Python environment.

References

1. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. (2008). https://bitcoin.org/bitcoin.pdf
2. Buterin, V.: A next-generation smart contract and decentralized application platform. Ethereum White Paper (2014). https://ethereum.org/en/whitepaper/
3. Hashim, F.A., Hussien, A.G., Mabrouk, M.S., Al-Atabany, W., Mirjalili, S., Abualigah, L., Rizk-Allah, R.M., Elaziz, M.A.: Snake Optimizer: A novel meta-heuristic optimization algorithm. Knowl. Based Syst. **242**, 108320 (2022). https://doi.org/10.1016/j.knosys.2022.108320
4. Al-Breiki, H., Refaei, M., Al-Rubaie, M., Jararweh, Y.: Blockchain scalability: challenges and solutions. Cluster Comput. **23**(4), 2089–2115 (2020). https://doi.org/10.1007/s10586-020-03004-0
5. Almasoud, A.S., Hussain, F.K., Hussain, O.K.: Systematic literature review of challenges in blockchain scalability. Appl. Sci. **11**(20), 9372 (2021). https://doi.org/10.3390/app11209372
6. Belotti, M., Bozzi, A., Christofaro, M., Bianchi, S.: A survey on blockchain scalability. IEEE Commun. Surv. Tutorials. **21**(3), 2886–2906 (2019). https://doi.org/10.1109/COMST.2019.2916805
7. Dabbagh, M., Kakavand, M., Tahir, M., Amphawan, A.: Performance analysis of blockchain platforms: empirical evaluation of Hyperledger Fabric and Ethereum. In *2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, pp. 1–6. (2020). https://doi.org/10.1109/IICAIET49801.2020.9257811
8. Baliga, A., Solanki, N., Verekar, S., Pednekar, P., Kamat, P., Chatterjee, S.: Performance characterization of blockchain consensus algorithms. In *2018 IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pp. 157–162. (2018). https://doi.org/10.1109/ICDCSW.2018.00036
9. Alharby, M.: Transaction latency within permissionless blockchains: analysis, improvement, and security considerations. J. Netw. Syst. Manage. **31**, 22 (2023). https://doi.org/10.1007/s10922-022-09717-w
10. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput. Surveys. **35**(3), 268–308 (2003)
11. Taher, S.S.H., Ameen, S.Y., Ahmed, J.A.: Enhancing blockchain scalability with Snake Optimization Algorithm: a novel approach. Front. Blockchain. **7**, 1361659 (2024). https://doi.org/10.3389/fbloc.2024.1361659
12. Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A.: Deep reinforcement learning: a brief survey. IEEE. Signal. Process. Mag. **34**(6), 26–38 (2017). https://doi.org/10.1109/MSP.2017.2743240
13. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 182–197 (2002). https://doi.org/10.1109/4235.996017
14. Herzel, A., Ruzika, S., Thielen, C.: Approximation methods for multiobjective optimization problems: a survey. INFORMS J. Comput. 1284–1299 (2021). https://doi.org/10.1287/ijoc.2020.1018
15. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction, 2nd edn. MIT Press (2018)

16. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bull. 80–83 (1945). https://doi.org/10.2307/3001968
17. Talbi, E.-G.: Metaheuristics: From design to implementation. Wiley (2009). https://doi.org/10.1002/9780470496916
18. Yang, X.-S.: Engineering optimization: An introduction with metaheuristic applications. Wiley (2010). https://doi.org/10.1002/9780470640425
19. Goldberg, D.E.: Genetic algorithms in search, optimization, and machine learning. Addison-Wesley (1989)
20. Kennedy, J., Eberhart, R.: A new optimizer using particle swarm theory. In Proceedings of the Sixth International Symposium on Micro Machine and Human Science (pp. 39–43). IEEE. (1995). https://doi.org/10.1109/MHS.1995.494215
21. Dorigo, M., Stützle, T.: Ant colony optimization. MIT Press (2004). https://doi.org/10.7551/mitpress/1290.001.0001
22. Storn, R., Price, K.: Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. 341–359 (1997). https://doi.org/10.1023/A:1008202821328
23. Kirkpatrick, S., Gelatt, C.D. Jr., Vecchi, M.P.: Optimization by simulated annealing. Science. **220**(4598), 671–680 (1983). https://doi.org/10.1126/science.220.4598.671
24. Mirjalili, S.: SCA: A sine cosine algorithm for solving optimization problems. Knowl. Based Syst. **96**, 120–133 (2016). https://doi.org/10.1016/j.knosys.2015.12.022
25. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014). https://doi.org/10.1016/j.advengsoft.2013.12.007
26. Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H.: Harris hawks optimization: Algorithm and applications. Future Generation Comput. Syst. **97**, 849–872 (2019). https://doi.org/10.1016/j.future.2019.02.028
27. Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., Gandomi, A.H.: The arithmetic optimization algorithm. Comput. Methods Appl. Mech. Eng. **376**, Article 113609. (2021). https://doi.org/10.1016/j.cma.2020.113609
28. Hashim, F.A., Houssein, E.H., Hussain, K., Mabrouk, M.S., Al-Atabany, W.: Honey badger algorithm: New metaheuristic algorithm for solving optimization problems. Math. Comput. Simul. **192**, 84–110 (2022). https://doi.org/10.1016/j.matcom.2021.08.013

Figures

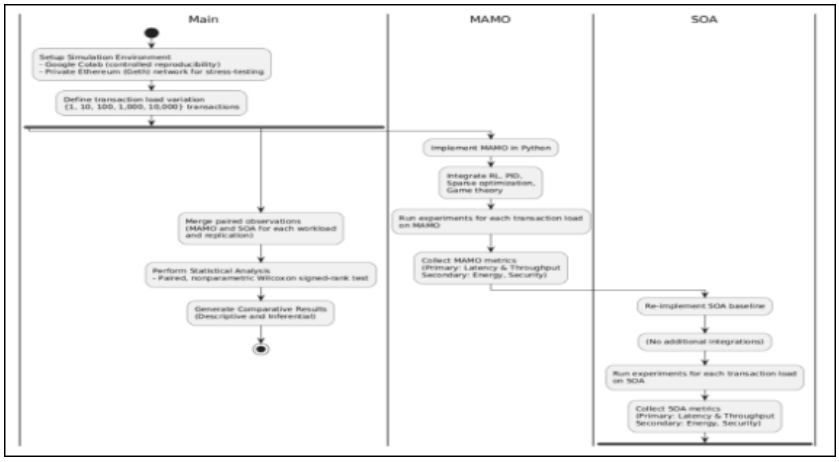


Figure 1

Overall Architecture of the Proposed MAMO Algorithm

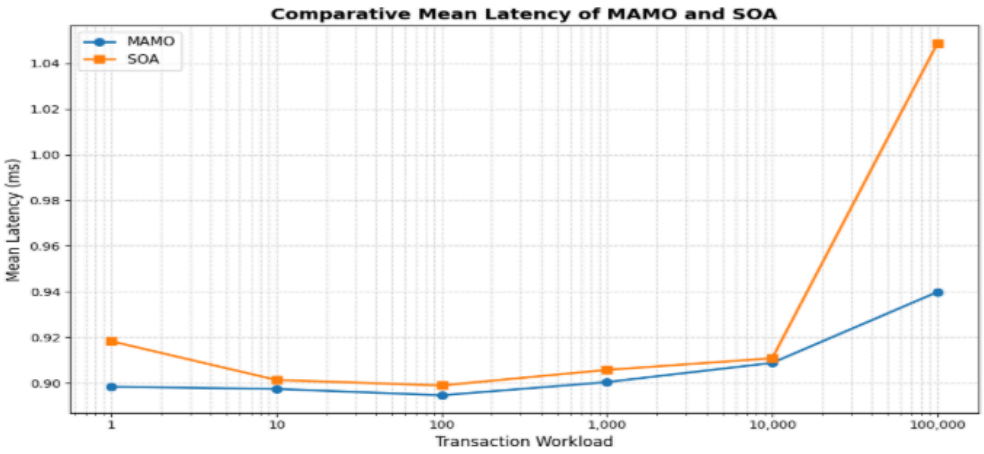


Figure 2

Latency Comparison

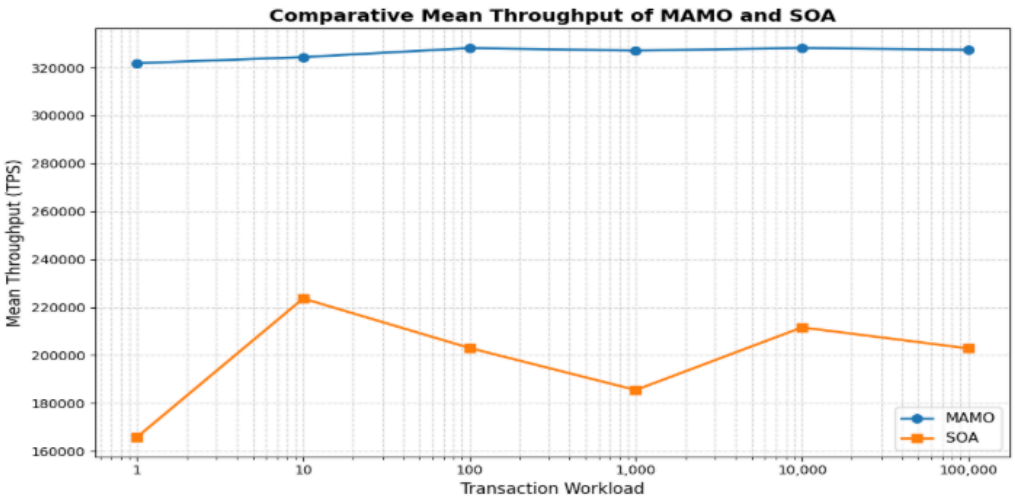


Figure 3

Throughput Comparison