Received 11 August 2022, accepted 13 September 2022, date of publication 23 September 2022,
date of current version 30 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3208702

RESEARCH ARTICLE

Black-Box for Blockchain Parameters Adjustment

VLADISLAV AMELIN[1], ERNEST GATIYATULLIN[2], NIKITA ROMANOV[1], RATMIR SAMARKHANOV[2], ROBERT VASILYEV[1], AND YURY YANOVICH[2,3], (Member, IEEE)
[1]GBC.AI Pty Ltd, Coomera, QLD 4209, Australia
[2]Faculty of Computer Science, HSE University, 109028 Moscow, Russia
[3]Center for Next Generation Wireless Technologies and IoT, Skolkovo Institute of Science and Technology, 121205 Moscow, Russia

Corresponding author: Yury Yanovich (y.yanovich@skoltech.ru)

This work was supported in part by the AWS Activate Program (https://aws.amazon.com/activate/?nc1=h_ls). The program with resources designed to help startups get started to use infrastructure needed to scale and science experiments.

ABSTRACT This paper introduces a function for blockchain performance evaluation as a black-box. The function runs the Solana blockchain test network with the only differences between the main network in a configuration file and the physical network to operate in. The black-box takes setup parameters as input, launches blockchain in a cloud, emulates artificial users' activity, and gives two outputs–transactions per second (tps) and drop rate. By default, the setup has six most important integer parameters and a network with three computers in the cloud, while one can vary eighty-nine parameters, the number of computers in the network and use local computers via black-box configuration files. The applied problem is to maximize the tps under a zero drop rate constraint. The black-box, like real blockchains, uses network communication, so reproducibility is an essential part of the design. We also provide an optimization baseline, showing the non-trivial results' reachability.

INDEX TERMS Blockchain, black-box, metaheuristic, surrogate optimization.

I. INTRODUCTION

Blockchains [1], [2] are peer-to-peer networks. In contrast to centralized networks, where an associated data storage updates by the will of a single peer, one can not update just one's local storage but should agree on the changes with other peers. The consensus is an agreement problem among many peers [3], [4]. Some peers may be faulty because of technical issues, own business or economic interests or hacker attacks. Consensus protocols in the presence of faulty peers exist since before blockchains [5] and play a key role in them. Blockchains group atomic changes of the associated data storage–transactions–into ordered batches–blocks–and achieve consensus about new blocks. Besides classic correctness requirements, blockchains need specific ones [6], [7], such as high performance regarding transactions per second (`tps`), fast transaction confirmation, and chain quality.

The fast transaction conformation is a measure of latency between a user broadcasting a transaction on the Internet and the transaction inclusion in an agreed block. If users generate transactions independent of the block generation events moments, the latency can not be less than half of the average block generation time. Furthermore, if the mempool–the set of the new correct transactions to be added to the blockchain–is small, the ongoing block will include the new transaction, and the resulting latency will be close to half of the block generation time. Blockchain needs to have a more significant throughput than the flow of new transactions to avoid the infinite growth of the mempool. The throughput regarding `tps` is a standard measure of blockchain performance.

The throughput and latency compete. On the one hand, one can have blocks without user transactions or one transaction each, then one can generate blocks according to the pre-agreed deterministic rule, and the average time between blocks can be negligible. Nevertheless, the throughput will be zero for empty blocks and small for one-transaction blocks. On the other hand, one can allow unlimited blocks and include all the transactions over a long period into a single block but generate blocks once per one hundred years. In this case, the overhead for consensus will be small, and one can achieve the maximum throughput. However, one will wait fifty years for transaction processing, which is infeasible for

The associate editor coordinating the review of this manuscript and approving it for publication was Mehdi Sookhak.

practical applications. So the blockchain has a Pareto front [8] of optimal operation regimes and needs to pick the proper regime through a trade-off between the current system needs. In practice, blockchains upper-bounds latency and maximizes `tps`. So do Bitcoin and Ethereum–the most prominent blockchains and cryptocurrencies by capitalization [9]. For example, on average, Bitcoin adjusts the difficulty parameter once per 2016 to have one block per 10 minutes. Ethereum picked an interval between blocks of 12 seconds based on Bitcoin statistical data to have enough time to synchronize a block over most peers [10].

Blockchain scalability is a hot topic, and the throughput increase is one of its dimensions [11], [12]. Low throughput results in higher market-driven transaction fees and higher latency at user activity peak [13], [14], finally, a bad user experience. Various approaches to speeding up the throughput exist. One idea is to change the consensus algorithm. Thus, EOS [15] shows around 4000 `tps` for latency less than 1 second with its implementation delegated proof of stake consensus protocol (DPoS) [16] compared to Bitcoin's 7 `tps` and Ehtereum's 14 `tps`. Alternative solutions show comparable results, with a theoretical limit of 700 000 `tps` with a bottleneck in Ethernet [17], [18], [19].

Another idea is to run N blockchains in parallel or make a hierarchical structure of blockchains to increase the throughput up to N times. Such multi-blockchain structures are called sidechain or parachain, and several solutions are proposed and implemented [20], [21]. A similar solution to the scalability problem pursued by Ethereum research is sharding protocols in which the whole blockchain network splits into different sections that can perform all necessary actions. This solution must be implemented in the base blockchain protocol itself and therefore requires reconstructing the whole network. These are called layer-1 solutions in blockchain and suffer from a trade-off between scalability and security; when we increase system performance by distributing transactions to shards, we decrease the computational resources for each of them [22], [23].

An alternative solution can be implemented at the user level and can significantly increase the performance of their applications. These solutions are called layer-2 solutions. One of them is organizing state channels by which users can interact with each other and place in a parent blockchain only the aggregated set of transactions. An example of such an approach includes lightning networks for Bitcoin, Counterfactual and Raiden for Ethereum [24], showing `tps` over 100 000. Another example of solutions of such kind is sharding protocols in which individual users check only part of incoming transactions. An example of such an approach includes Plasma for Ethereum [25], [26]. The main difference between the two layer-2 approaches is that in the state channels, transactions are validated utilizing a consentient consensus among participants. In contrast, in shading protocols, transactions are validated by a Merkle proof submitted to the main chain by sharding users, which leads to a delay in the transaction approval process.

We currently have networks without access to change parameters for research for a fixed blockchain architecture. Nevertheless, the source code of the blockchain is publicly available. So we can launch a test system in our environment and vary parameters as we wish. Both network and protocol parameters define the blockchain operation regime. Some of the network parameters are observable, for example, the current pool of unconfirmed transactions; some of the network parameters are not observable (latent), for example, blockchain network graph, connection latency, and bandwidth [27], [28]. Unobservable parameters play an essential role in the performance; for example, average propagation delay affects transaction latency, and it is helpful to estimate them with a particular accuracy [29]. Latent parameters result in the impossibility of emulating the network with the test network but only simulating.

This paper is a continuation of the research [30], where we presented a machine learning view on blockchain parameters adjustment, designed a Solana blockchain-based testbed, and performed a sensitivity analysis of the available parameters. This paper presents a function for blockchain performance evaluation in the form of an interactive black-box. The function runs the Solana blockchain test network with the only differences between the network in a configuration file and the physical network to operate in. The main objective is to maximize the throughput, while more problems, including the results propagation to the network, are of interest too. The rest of the paper is organized as follows. Section 2 views the Solana blockchain as a black-box for parameter adjustment. Section 3 introduces the developed black-box and details its reproducibility. The optimization with a limited black-box computation budget shows the reachability of non-trivial results in Section 4. Finally, Section 5 concludes the paper.

II. SOLANA BLOCKCHAIN AS BLACK-BOX

Solana is a high-performance permissionless blockchain [18], [31]. Its network consists of clusters–sets of validators working together to serve client transactions and maintain the ledger's integrity. Clusters may coexist. When two clusters share a common genesis block, they attempt to converge. For example, one set of clusters with a common genesis block is a Solana network, whose token is in the top ten cryptocurrencies by market capitalization as of July 2022 [9]. Otherwise, each cluster ignores the existence of the others with different genesis blocks. One sets a genesis configuration to create a cluster and runs a validator. Additional validators then register with any registered member of the cluster.

Clients send transactions to any validator's Transaction Processing Unit (TPU) port. A validator forwards the transaction to the designated leader. If the validator is in the lead role, the node bundles incoming transactions, timestamps them, creating an entry, and pushes them onto the cluster's data plane. Once on the data plane, the transactions are validated by validator nodes, effectively appending them to the ledger. Solana defines confirmation as the duration of time from the leader timestamps a new transaction to the recognition of a

TABLE 1. The most important Solana's parameters for `tps`.

Parameter	Default value	Description
`NUM_THREADS`	4	To improve the throughput to the database, Solana supports connection pooling using multiple threads, each maintaining a connection to the PostgreSQL database.
`TICKS_PER_SLOT`	63	The period for which each leader ingests transactions and produces a block is called a slot. Each slot is divided into logical units called ticks.
`RECV_BATCH_MAX_CPU`	1000	Transaction signature verification is a repetitive and time-consuming operation. Solana sends a batch of transactions for the verification to schedule the load. By default, a CPU verifies transactions, and `RECV_BATCH_MAX_CPU` defines the maximum batch size.
`ITER_BATCH_SIZE`	1000	Batch inserts into a database reduce the round trips to the database and, hence, improve performance under high load.
`HASHES_PER_SECOND`	2000000	Solana documentation states Google Cloud Platform's n1-standard hardware to be the standard hardware and claims it to have `HASHES_PER_SECOND` hash computations per second.
`TICKS_PER_SECOND`	160	A ledger entry that estimates wallclock duration is called a tick.

supermajority of ledger votes. The blockchain rotates leaders at fixed intervals, called slots. Each leader produces entries during its allotted slot only.

The proof-of-Stake consensus protocol allows for achieving consensus in Solana. The Proof-of-History (PoH) technique guarantees reliable transaction ordering, which allows synchronization with block parts rather than whole blocks. PoH is not a consensus mechanism but an application-specific verifiable-delay function (VDF). The verification in PoH has the same asymptotic as the proposal creation but with a better constant, while the verification in VDF has faster rates. Another difference between PoH and VDFs is that a VDF tracks duration only. PoH includes hashes of any data the application observed as well.

Due to the Solana design, the ratio of sent transactions and successfully written in block, i.e. finalized, maybe not equal 1. The per cent of dropped transactions through a time interval is called *drop rate*. The *drop rate* and transaction per second (*tps*) are the main pair to characterize Solana performance like latency and *tps* for Bitcoin and Ethereum. Note that *tps* counts only successful transactions.

The high-load stationary [32] operating regimes are of interest. We achieve them by launching a cluster on our infrastructure–a test network (testnet)–and running a benchmark with a big batch of user transactions. The benchmark outputs are *tps* and *drop rate*. As a result, we consider the whole workflow as a function $y = F(x, \theta, \xi)$, where

- x is the vector of observable uncontrollable blockchain parameters. For example, the number of nodes is a component of x.
- θ is the vector of (observable) controllable blockchain parameters. For example, block size is a component of θ. In general it may depends from uncontrollable parameters $\theta(x)$.
- ξ is the latent (uncontrollable) blockchain parameters. For example, nodes' computational power is a component of ξ.
- y is the vector of the blockchain operation regime with given parameters (x, θ, ξ). In the benchmark, *tps* and *drop rate* are the only components of y. As the *tps* is non-negative, we formally set $tps = 0$ for the non-zero *drop rate* cases. After this modification, y becomes a one-dimensional vector, i.e., scalar.

The uncontrollable parameters $x = x_0$ are constant for a given testnet. We treat latent parameters ξ as independent random variables for different runs. The testnet takes around ten minutes to evaluate a random function $y = f(\theta) = F(x_0, \theta, \xi)$. Under computational time constraint and lack of an analytic model for the function f, a black-box optimization problem arises: maximize the mean value of $f(\theta)$ over θ with a maximum $N_{\max}$ calls of f.

III. BLACK-BOX DESCRIPTION

A. STRUCTURE

Our Solana blockchain black-box is openly available on Github [33] under Apache 2.0 licence. The core of the black-box is a fork of Solana version 1.5.0. By default, Solana does not allow to change parameters. So we modified it to support 89 controllable user-input parameters θ. One can change all these parameters for the black-box, although we use 6the most important parameters [30] by default. Table 1 lists the parameters, their meaning, and default values. All the parameters are unsigned integers in Solana because of the implementation. In theory, some parameters are real numbers, for example, `HASHES_PER_SECOND`. At the same time, others are fundamentally integers, for example, `NUM_THREADS`. To start evaluation, one must fill the credential file and run the black-box as a Python function (see the Examples folder in the repository). The function takes 6 integer inputs as θ from a $\pm 15\%$ box around corresponding Solana's default values $\theta_0 = (4, 1000, 1000, 63, 2000000, 160)$. Other words, $\theta \in \Theta \subset \mathbb{Z}^6$, where

$$\Theta = \Big([3, 5] \times [850, 1150] \times [850, 1150] \times [53, 73] \times [1700000, 2300000] \times [136, 184]\Big) \cap \mathbb{Z}^6 \qquad (1)$$

and $\mathbb{Z}^6$ denotes the set of all six-dimensional integers. The output is two-dimensional: `tps` and `drop rate`. The applied problem is to maximize `tps` under the zero *drop rate* constraint.

The black-box is a computational job as follows (see Figure 1)

1) Allocate computers within a single data centre. By default, we use four `t2.2xlarge` virtual

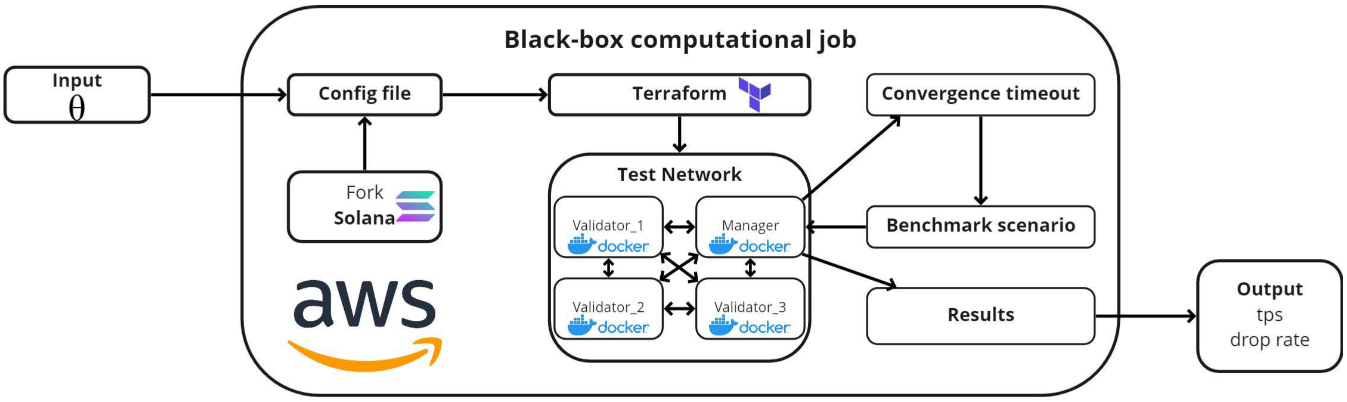


FIGURE 1. **Black-box structure.**

instances in Amazon Elastic Compute Cloud (EC2). Three instances are for the Solana cluster, and one is to manage the performance test. We use this data centre computer configuration because of its widespread availability. However, one can edit the configuration file to change a computer's type, the data centre or a cloud platform.

2) Start Solana test cluster on instances with given parameters plus one instance to manage the experiment. All the instances are from the same Docker container but with different Terraform configurations.
3) Wait until the nodes start communication. The cluster needs to allocate resources and synchronize communication upon start. Solana calls it a network convergence. Benchmark results may be unstable before the synchronization. We numerically estimate the convergence time in Section III-B.
4) Run Solana's `bench-tps` benchmark. The manager instance generates a million user accounts, fills them with coins and sends transactions to simulate user activity. Test cluster processes the transactions; each node generates log files. Manager instance collects logs, computes experimental results, and returns the black-box output. The black-box job is done.

B. NETWORK CONVERGENCE

Upon the black-box launch, a testnet and a manager instance start. Each computer in the testnet needs to allocate resources and synchronize the consensus-driven communication. The manager instance creates user accounts to send a transaction from them. These processes take time, and the Solana team recommends waiting until the network converges [31].

To estimate the system convergence time numerically, we performed the following experiment. For the Solana's default point $\theta_0 = (4, 63, 1000, 1000, 2000000, 160)$ and per each waiting time between network start up and benchmark run from 0 to 90 seconds with step 10 seconds, the black-box has been computed 10 times. The mean *tps* and its standard deviation (std) as functions of the waiting time are in Figure 2.

The reason for a performance peak of around 30 seconds is unknown. Such effect is not present on a local testnet but only in the cloud. So the cloud resource allocation mechanism may be the clue. Delays ≥ 70 seconds show stable stationary results. We set the delay of 80 seconds as a default timeout in a black-box for network convergence, but one can set his value as an optional black-box parameter.

C. SIGNAL-TO-NOISE RATIO

The function $f(\theta) = F(x_0, \theta, \xi)$ is random. Signal-to-noise ratio ($\mathtt{SNR}$) as a measure that compares the level of a desired signal to the level of background noise [34]

$$\mathtt{SNR} = \frac{\mathbf{var}_\theta \mathbf{E}_\xi F(x_0, \theta, \xi)}{\mathbf{E}_\theta \mathbf{var}_\xi F(x_0, \theta, \xi)}, \tag{2}$$

where $\mathbf{E}$ is the mathematical expectation and $\mathbf{var}$ is the variance. To estimate $\mathtt{SNR}$, we generated a random sample $\{\theta_n\}_{n=1}^N$ of $N = 10$ independent uniformly distributed within the allowed domain points Θ (1). The function $F(x_0, \theta, \xi)$ was evaluated $M = 10$ times in each point from $\{\theta_n\}_{n=1}^N$, resulting in a dataset $\left\{F(x_0, \theta_n, \xi_{n,m})\right\}_{n,m=1}^{N,M}$. Two indexes of ξ show that $\xi_{n,m}$ are independent for different pairs of (n, m). The delta method [35] estimation of $\mathtt{SNR}$ (2) equals

$$\hat{\mathtt{SNR}} = \frac{\overline{\left(\overline{F(x_0, \theta_n, \xi_{n,m})}_m - \overline{F(x_0, \theta_n, \xi_{n,m})}_{n,m}\right)^2}_n}{\overline{\left(\overline{\left(\overline{F(x_0, \theta_n, \xi_{n,m})}_m - F(x_0, \theta_n, \xi_{n,m})\right)^2}_m\right)}_n}, \tag{3}$$

where $\overline{A(i)}_i = \frac{1}{I}\sum_{i=1}^{I} A(i)$ is an average of $A(i)$ over index i from 1 to I. For example,

$$\overline{F(x_0, \theta_n, \xi_{n,m})}_m = \frac{1}{M}\sum_{m=1}^{M} F(x_0, \theta_n, \xi_{n,m})$$

and

$$\overline{F(x_0, \theta_n, \xi_{n,m})}_{n,m} = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{M}\sum_{m=1}^{M} F(x_0, \theta_n, \xi_{n,m})\right).$$

The resulting value of $\hat{\mathtt{SNR}} = 15.6$ or, expressed using the logarithmic decibel scale,

$$\hat{\mathtt{SNR}}_{\mathtt{db}} = 10 \cdot \log_{10} \hat{\mathtt{SNR}} = 11.9,$$

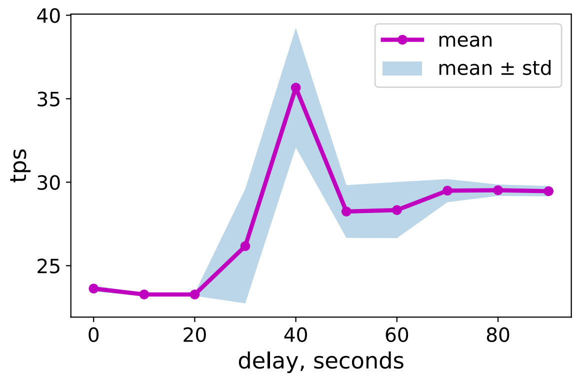


FIGURE 2. ***tps* as a function of delay.**

which indicates a low relative level of noise, and treated in telecommunications as an accepted minimum to establish an unreliable connection.

IV. OPTIMIZATION BASELINE

We considered three baseline optimization methods, namely

- `Random`: generate a random (space filling [36]) dataset within a computational budget, evaluate the black-box in each point, return the maximum value as an optimum.
- `Metaheuristic`: use optimization methods that orchestrate an interaction between local improvement procedures and higher level strategies and capable to deal with discrete input variables [37].
- `Surrogate`: generate a small initial design, perform search steps with a pre-trained statistical model–surrogate–instead of expensive black-box, iteratively extend the design within computational budget and update the surrogate model [38], [39].

The inputs are integers, which limits the methods choice. The used three methods are not comprehensive but only to achieve non-trivial results.

A. SPACE FILLING DESIGNS

The design is a set of input points. Model-free designs aim to represent the diversity of an unknown function in a given domain without information about the models we intend to use with them. Finite nets are good candidates for helping to understand the form of the unknown function if the function is Lipschitz continuous or smoother. The design of practical finite nets is a problematic task, even in a hypercube domain. For example, one can take a uniform grid with k levels per coordinate and consider all combinations of levels as a finite net design–full factorial design. The number of points in the full factorial design is k^d, where d is dimensionality, which is impractical for neither large k nor d. The full factorial design supports only a few design sizes for each dimensionality, i.e., one can not generate 100 point design in a 6-dimensional space.

Random designs are easy to generate. However, the space-filling parameter ε is difficult to compute in an arbitrary unstructured design, and some points may be close because of stochastic effects. To avoid too close points by design, we can split each coordinate into bins, and only one value per bin is chosen for the design, or with the smallest possible repeats number for integer coordinates with a few values in the design domain. These random designs are called Latin HyperCubes (LHC). Instead of the ε estimation, a pair-wise distance criterion is optimized, for example, maximin pair-wise distance. The design is optimal LHC if it optimizes the chosen criteria. If the symmetrical one about the centre is in the design for each point in the design, too, the LHC is called symmetric (SLHC). SLHC, as a subtype of the orthogonal design, ensures zero correlation among estimation of linear effects.

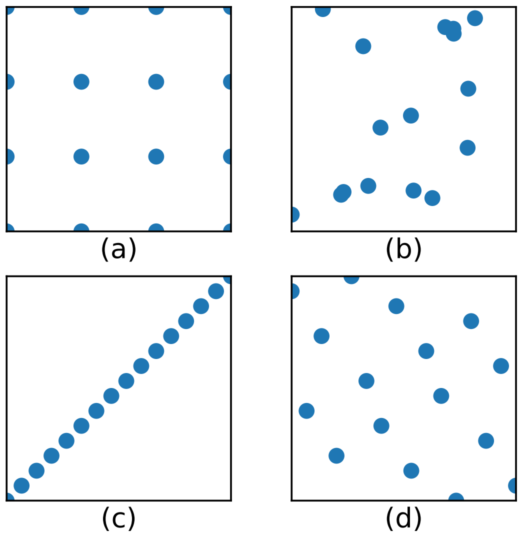


FIGURE 3. Examples of designs in two-dimensional space with $N = 16$ points. Left to right, top to bottom: (a) full factorial, (b) random uniform, (c) bad SLHC, (d) optimal SLHS. One dot represents one point of the design.

The examples of full factorial, uniform random, bad SLHC and optimal SLHC are in Figure 3. We use Python implementation of optimal SLHC [40] from library PySOT [41] in our experiments as `Random`.

B. METAHEURISTIC OPTIMIZATION

Metaheuristics are solution methods that orchestrate an interaction between local improvement procedures and higher-level strategies to perform a robust search of solution space [37]. Metaheuristic algorithms have a very little theoretical justification for being global optimizers, but are widely used in practice because of good results. A wide class of metaheuristic algorithms are inspired by the evolutionary computation. And the reason for their good properties can be the utilization of the solution candidates' population to get the better one and diversity control for sustainable search.

The Python implementation [42] of Multi-Verse Optimizer (MVO) [43]–a cosmology inspired evolutionary algorithm for global optimization–is used in the experiment as `Metaheuristic`. MVO starts with creating a set of random solution candidate points–universes. The set updates iteratively–the universes evolve over the time. A coordinate-wise substitution and a random homothety with the best current solution as a fixed point are modifications through iterations. Sending a coordinate to a pool for substitution is a falling into a black hole and receiving a coordinate from the pool is an outgoing from a white hole. Homothety is a wormhole to transport its objects through space.

C. SURROGATE OPTIMIZATION

A surrogate is an explicit function that approximates another function [39]. Compared to the real-world or computational simulation, the surrogate model is easy to compute and has an analytic form for derivatives. The surrogate model requires a dataset to start with. A space filling design, for example, SLHC, is a common option to compute the initial surrogate model. An algorithm for the surrogate model construction is defined by an input space dimensionality Θ (1) and the dataset size, which is upper bounded by the black-box computation budget. Non-parametric kernel methods are standard for small datasets [44], [45]. One of them–radial basis functions (RBF)–approximate the black-box in our case.

A surrogate optimization uses the cheapness of the surrogate model in an iterative procedure: evaluate surrogate model many times to find a suitable candidate point for an optimum, evaluate the black-box in the candidate point, update the surrogate model based on the new black-box output and repeat the steps until the black-box evaluation budget is over. The surrogate model is an approximation of the black-box. So, the optima of the surrogate model is not the optima of the black-box because of approximation error. The suitable candidate point is not directly an optimum of the surrogate model, but a trade-off between the surrogate model optima and the surrogate model uncertainty decrease. For example, the DYnamic COordinate search using Response Surface (DYCORS) [46] algorithm computes the next evaluation point based on a score, which equals to the weighted sum of the candidate's optimality and distance from the dataset. The dynamic coordinate search with perturbations defines the set of candidates. The perturbation probability decreases as the black-box evaluations increase.

We use the Python library PySOT [41] with SLHC for initial design, RBF for surrogate model and DYCORS strategy in the experiment as `Surrogate`.

D. NUMERICAL RESULTS

We run `Random`, `Metaheuristic` and `Surrogate` optimization methods with the black-box evaluation budget $N_{\max} = 100$ to check if they increase the `tps` for a fixed Solana cluster configuration. The experiments Python script is available in the repository [33]. The results are in Figure 4. All the methods achieved `tps` in the interval (39, 40) from the initial 29, and `Surrogate` has shown the fastest convergence: 20 black-box evaluations compared to 60 for `Random` and `Metaheuristic`.

As the black-box is noisy, we also evaluated 10 times at each `Random`, `Metaheuristic` and `Surrogate` optimal point, computed mean and standard deviation and compared them with the values at the `Default` point. The results are in Table 2. The mean values of the solutions are from 38.1 to 38.9 with the biggest value for `Surrogate`. The standard deviation varies from 1.3 for `Random` to 0.8 for `Surrogate`. The results are statistically significantly greater than the values for the `Default` by the empirical (three-sigma) rule for all three optimization approaches.

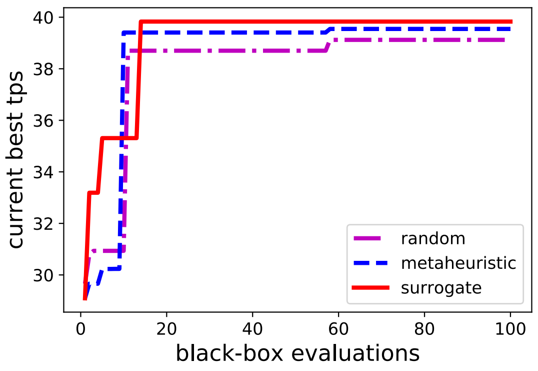


FIGURE 4. Current best `tps` as a function of the black-box evaluation iteration.

TABLE 2. Optimization results.

Method	θ	`tps`
`Default`	(4, 1000, 1000, 63, 2000000, 160)	29.5 ± 0.4
`Random`	(3, 1033, 867, 67, 1951151, 142)	38.1 ± 1.3
`Metaheuristic`	(4, 1082, 929, 67, 1754909, 136)	38.6 ± 0.9
`Surrogate`	(3, 980, 859, 68, 2007222, 136)	38.9 ± 0.8

The standard deviation depends on the point, so the random noise is not stationary over the domain Θ (1). Optimal points by `Random`, `Metaheuristic` and `Surrogate` are far from each other, which shows either a locality for extrema or a big impact of the noise.

V. CONCLUSION AND FUTURE WORK

Blockchain architectures keep various parameters constant during their life without a justification for default values. We have the mainnet without access to change parameters for research and publicly available source code for a fixed system. So one can launch a testnet in one's environment and vary parameters as one wishes. Uncontrollable latent parameters introduce randomness and make it impossible to emulate the mainnet with the testnet but only simulate it. The testnet can reveal the blockchain architecture behaviour, check the operation regime sensitivity to the parameters chosen, justify the default parameters, and propagate the results to the mainnet.

This paper introduces a Solana blockchain testnet as a black-box function. By default, the black-box takes six the most important parameters as input and returns two blockchain performance parameters, namely, `tps` and `drop rate`. The experiments show that the black-box noise is significantly smaller than the output variation over the input domain. The black-box is still not reproducible but has an acceptable signal-to-noise ratio (SNR). The high SNR is achieved by launching the black-box within a single data centre on unified hardware with an internal timeout between the start-up and performance measurements to allow network convergence. To increase SNR further, one can logically isolate the computation environment or migrate to a physically isolated local network.

We used three baseline optimization techniques and all of them has shown 30% `tps` increase under `drop rate` $= 0$ constraint. So, the parameter choice has a real impact on the performance. Validators are free to join and leave permissionless blockchains, so the network configurations change over time. As the optimal parameters can be different for different configurations, we treat an adaptive parameter adjustment algorithm as future long-distance work. Some parameters are only local (`NUM_THREADS` and `HASHES_PER_SECOND` for Solana) and can be changed at any time, while some parameters affect the blockchain protocol (`TICKS_PER_SLOT` and `TICKS_PER_SECOND` for Solana) and change through consensus only. As a result, the adaptive parameter adjustment algorithm has a validator (local) and a protocol (global) parts.

The experiments revealed the noise to be non-stationary. From the statistical point of view, the black-box noise model needs clarification. We listed machine learning tasks for blockchain parameter adjustment in the paper [30], and they entirely apply to the proposed black-box from feature importance and sensitivity analysis to data fusion and change point detection. Finally, the Solana blockchain testnet launched in an Amazon cloud is only a model example, while the proposed approach allows other blockchains and infrastructure.

REFERENCES

[1] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] V. Buterin. (Jan. 2014). *Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform*. Ethereum. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[3] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982. [Online]. http://portal.acm.org/citation.cfm?doid=357172.357176

[4] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *J. ACM*, vol. 35, no. 2, pp. 288–323, 1988. [Online]. Available: http://portal.acm.org/citation.cfm?doid=42282.42283

[5] J. Dollimore, T. Kindberg, and G. Coulouris, *Distributed Systems: Concepts and Design*. Reading, MA, USA: Addison-Wesley, 2005.

[6] Y. Yanovich, I. Ivashchenko, A. Ostrovsky, A. Shevchenko, and A. Sidorov. (2018). *Exonum: Byzantine Fault Tolerant Protocol for Blockchains*. [Online]. Available: https://bitfury.com/content/downloads/wp-consensus-181227.pdf

[7] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing hyperledger fabric blockchain platform," in *Proc. IEEE 26th Int. Symp. Model., Anal., Simul. Comput. Telecommun. Syst. (MASCOTS)*, Sep. 2018, pp. 264–276. [Online]. Available: https://ieeexplore.ieee.org/document/8526892/

[8] K. Deb, "Multi-objective optimization," in *Search Methodologies*. Boston, MA, USA: Springer, 2014, pp. 403–449, doi: 10.1007/978-1-4614-6940-7_15.

[9] (2022). CoinMarketCap. *Dash Price, Charts, Market Cap, and Other Metrics*. [Online]. Available: https://coinmarketcap.com/currencies/dash/

[10] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Proc. IEEE P2P*, Sep. 2013, pp. 1–10. [Online]. Available: http://ieeexplore.ieee.org/document/6688704/

[11] V. Buterin. (2021). *The Limits to Blockchain Scalability*. [Online]. Available: https://vitalik.ca/general/2021/05/23/scaling.html

[12] M. H. Nasir, J. Arshad, M. M. Khan, M. Fatima, K. Salah, and R. Jayaraman, "Scalable blockchains—A systematic review," *Future Gener. Comput. Syst.*, vol. 126, pp. 136–162, Jan. 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167739X21002971

[13] G. A. Pierro and H. Rocha, "The influence factors on ethereum transaction fees," in *Proc. IEEE/ACM 2nd Int. Workshop Emerg. Trends Softw. Eng. Blockchain (WETSEB)*, May 2019, pp. 24–31. [Online]. Available: https://ieeexplore.ieee.org/document/8823908/

[14] A. Donmez and A. Karaivanov, "Transaction fee economics in the ethereum blockchain," *Econ. Inquiry*, vol. 60, no. 1, pp. 265–292, Jan. 2022, doi: 10.1111/ecin.13025.

[15] (2017). I. Grigg. *EOS—An Introduction*. [Online]. Available: https://eos.io/documents/EOS_An_Introduction.pdf

[16] B. Xu, D. Luthra, Z. Cole, and N. Blakely. (2018). *EOS: An Architectural, Performance, and Economic Analysis*. [Online]. Available: https://www.whiteblock.io/library/eos-test-report.pdf

[17] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 10401. Cham, Switzerland: Springer, 2017, pp. 357–388. [Online]. Available: https://assets.ctfassets.net/r1dr6vzfxhev

[18] A. Yakovenko. (2018). *Solana: A New Architecture for a High Performance Blockchain*. [Online]. Available: https://solana.com/solana-whitepaper.pdf

[19] (2020). Ethereum. *Ethereum 2.0 Specifications*. [Online]. Available: https://github.com/ethereum/eth2.0-specs

[20] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. K. Miller, A. Poelstra, and J. T. Corallo. (2014). *Enabling Blockchain Innovations With Pegged Sidechains*. [Online]. Available: https://pdfs.semanticscholar.org/1b23/cd2050d5000c05e1da3c9997b308ad5b7903.pdf

[21] G. Wood. (2017). *Polkadot: Vision for a Heterogeneous Multi-Chain Framework*. Whitepaper. [Online]. Available: https://github.com/w3f/polkadot-white-paper/raw/master/PolkaDotPaper.pdf

[22] V. Buterin. (2021). *Sharding-FAQs | Ethereum Wiki*. [Online]. Available: https://eth.wiki/sharding/Sharding-FAQs

[23] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Oct. 2016, pp. 17–30. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2976749.2978389

[24] J. Coleman, L. Horne, and L. Xuanji. (2018). *Counterfactual: Generalized State Channels*. [Online]. Available: https://l4.ventures/papers/statechannels.pdf

[25] P. Prihodko, S. Zhigulin, M. Sahno, and A. Ostrovskiy. (2016). *Flare: An Approach to Routing in Lightning Network*. [Online]. Available: https://bitfury.com/content/downloads/whitepaper_flare_an_approach_to_routing_in_lightning_network_7_7_2016.pdf

[26] J. Poon and V. Buterin. (2017). *Plasma: Scalable Autonomous Smart Contracts*. White Paper. [Online]. Available: https://plasma.io/plasma.pdf

[27] V. Deshpande, H. Badis, and L. George, "BTCmap: Mapping bitcoin peer-to-peer network topology," in *Proc. IFIP/IEEE Int. Conf. Perform. Eval. Modeling Wired Wireless Netw. (PEMWN)*, Sep. 2018, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8548904/

[28] G. Fanti, S. B. Venkatakrishnan, S. Bakshi, B. Denby, S. Bhargava, A. Miller, and P. Viswanath, "Dandelion++," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 2, no. 2, pp. 1–35, Jun. 2018, doi: 10.1145/3224424.

[29] X. Xu, G. Sun, L. Luo, H. Cao, H. Yu, and A. V. Vasilakos, "Latency performance modeling and analysis for hyperledger fabric blockchain network," *Inf. Process. Manage.*, vol. 58, no. 1, Jan. 2021, Art. no. 102436. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0306457320309298

[30] V. Amelin, N. Romanov, R. Vasilyev, R. Shvets, Y. Yanovich, and V. Zhygulin, "Machine learning view on blockchain parameter adjustment," in *Proc. 3rd Blockchain Internet Things Conf.*, New York, NY, USA, Jul. 2021, pp. 38–43, doi: 10.1145/3475992.3475998.

[31] (2018). Solana Foundation. *Solana Docs*. [Online]. Available: https://docs.solana.com/

[32] R. D. Yates and D. J. Goodman, *Probability and Stochastic Processes: A Friendly Introduction for Electrical and Computer Engineers*, 3rd ed. New York, NY, USA: Wiley, 2014.

[33] (2022). GBC.AI. *Github Repository: Black-Box for Blockchain Parameters Adjustment*. [Online]. Available: https://github.com/GBC-AI/black-box

[34] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. London, U.K.: Pearson, 2008.

[35] L. Wasserman, *All of Nonparametric Statistics* (Springer Texts in Statistics). New York, NY, USA: Springer, 2006, doi: 10.1007/0-387-30623-4.
[36] A. D. Keedwell and J. Dénes, *Latin Squares and Their Applications*. Sydney, NSW, Australia: Elsevier, 2015. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/C20140034120
[37] M. Gendreau and J.-Y. Potvin, *Handbook of Metaheuristics* (International Series in Operations Research & Management Science), vol. 272, 3rd ed. Cham, Switzerland: Springer, 2019, doi: 10.1007/978-3-319-91086-4.
[38] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathna, and P. K. Tucker, "Surrogate-based analysis and optimization," *Prog. Aerosp. Sci.*, vol. 41, no. 1, pp. 1–28, Jan. 2005. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0376042105000102
[39] S. Koziel and L. Leifsson, *Surrogate-Based Modeling and Optimization*. New York, NY, USA: Springer, 2013, doi: 10.1007/978-1-4614-7551-4.
[40] K. Q. Ye, W. Li, and A. Sudjianto, "Algorithmic construction of optimal symmetric Latin hypercube designs," *J. Stat. Planning Inference*, vol. 90, no. 1, pp. 145–159, Sep. 2000.
[41] D. Eriksson, D. Bindel, and C. A. Shoemaker, "PySOT and POAP: An event-driven asynchronous framework for surrogate optimization," 2019, *arXiv:1908.00420*.
[42] N. Van Thieu, "A collection of the state-of-the-art meta-heuristics algorithms in Python: Mealpy," 2020, doi: 10.5281/zenodo.3711948.
[43] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, "Multi-verse optimizer: A nature-inspired algorithm for global optimization," *Neural Comput. Appl.*, vol. 27, no. 2, pp. 495–513, Feb. 2016, doi: 10.1007/s00521-015-1870-7.
[44] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006.
[45] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Springer Series in Statistics). New York, NY, USA: Springer, 2009.
[46] R. G. Regis and C. A. Shoemaker, "Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization," *Eng. Optim.*, vol. 45, no. 5, pp. 529–555, 2013, doi: 10.1080/0305215X.2012.687731.

VLADISLAV AMELIN received the bachelor's degree from the Moscow Institute of Electronic Engineering, National Research University, and the master's degree from the Department of Mathematical Forecasting Methods, Faculty of Computational Mathematics and Cybernetics, Moscow State University.

He is currently an Architecture and Machine Learning Expert. He is a Prize-Winner of international competitions, student's Olympic, and University games in mathematics and programming (2013–2016). He developed the model for predicting the outflow of bank clients and the system of interaction with clients to retain and increase loyalty (based on the list of financial products offered by the bank and the possibilities of their adaptation to a particular client). He has an extensive background in developing innovative products and vast experience in technical team management and internal business processes. He successfully implemented AI products and services in healthcare, fintech, retail, and blockchain. He is currently the Chief Technical Officer at GBC.AI Pty Ltd. In this role, he is the author of solutions based on AI and blockchain technologies. His research interests include machine learning, deep learning, machine vision, blockchain, consensus protocols, privacy, and applications.

ERNEST GATIYATULLIN received the B.S. degree in nanotechnology from Kazan Federal University, Kazan, Russia. He is currently pursuing the master's degree with HSE University.

He worked as a Laboratory Assistant with the Neurobiology Laboratory, Kazan Federal University, where he worked on the problem of bionic prosthesis controlling with a machine learning model, trained on the signals of arm muscles' contractions. His current research interests include machine learning and artificial intelligence.

NIKITA ROMANOV received the master's degree from the Department of Mathematical Forecasting Methods, Faculty of Computational Mathematics and Cybernetics, Moscow State University.

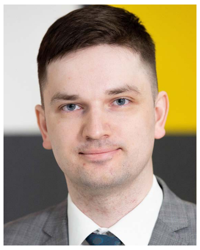
He is currently a Software Engineer for neural network technologies and machine learning. His most notable projects are the development and construction of a model for predicting the occurrence of receivables of counterparties for several months in advance, the development of an anomaly detection model for industrial enterprises (search for deviations in multidimensional time series with discrete signals), development of models for predicting the amount of traffic and search for anomalies in the network of retail stores, and development of a model for detecting and localizing road pavement defects by video recording using neural network algorithms. He is currently the Chief Data Officer at GBC.AI Pty Ltd. In this role, he is the author of solutions based on AI and blockchain technologies. His research interests include machine learning, deep learning, blockchain, consensus protocols, privacy, natural language processing, DevOps, and simulation.

RATMIR SAMARKHANOV received the B.S. degree in informatics and computer engineering from the Moscow Aviation Institute, Russia, in 2021. He is currently pursuing the M.S. degree in data science with HSE University, Moscow.

He is also working as Software Engineer with HSE University. His research interests include software and computer architecture, deep learning, and artificial intelligence.

ROBERT VASILYEV received the bachelor's degree from the Moscow Institute of Electronic Engineering, National Research University, the master's degree in economics from the Russian Presidential Academy (RANEPA), and the master's degree in applied physics and mathematics from the Moscow Institute of Physics and Technology, where he is currently pursuing the master's degree.

He is currently the Head of the AI Laboratory and the Vice President of the Artificial Intelligence Laboratories Association. He is an Expert of the Digital Economy Working Group in the field of big data and blockchain, the Prize-Winner of international student competitions in microelectronics (MIEE 2015) and physics (MIEE 2011), and the Winner of start-up competitions in the development and launch of the project "Digital model of the semiconductor industry." He is currently the Chief AI Officer at GBC.AI Pty Ltd. In this role, he is the author of AI and blockchain technologies solutions. His research interests include machine learning, deep learning, machine vision, blockchain, deep tech development team management, applied AI tech, data science, cloud computing, and applications.



YURY YANOVICH (Member, IEEE) received the bachelor's and master's degrees (Hons.) in applied physics and mathematics from the Moscow Institute of Physics and Technology, Moscow, Russia, in 2010 and 2012, respectively, and the Ph.D. degree in probability theory and mathematical statistics from the Institute for Information Transmission Problems, Moscow, in 2017.

He is currently a Senior Research Scientist at the Skolkovo Institute of Science and Technology, Moscow. He is the author of Exonum consensus protocol and has been a Lecturer in the introduction to blockchain course at top Russian universities, since 2017. His research interests include blockchain, consensus protocols, privacy, and applications.