Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.777932

BLOCKCHAIN SIMULATOR OPTIMIZATION USING GENETIC ALGORITHMS

Douaa H. Khalaf [1], Muntaser A. Salman[2*] and Rabah N. Farhan[3]

[1] Department of Computer Science, College of Computer Science and Information Technology, University of Anbar, Province of Anbar, Iraq.

[2] Department of Information System, College of Computer Science and Information Technology, University of Anbar, Province of Anbar, Iraq.

[3] Department of Control and Communications, Renewable Energy Research Center, University of Anbar, Province of Anbar, Iraq.

[*]Corresponding author: dou20c1006@uoanbar.edu.iq

E-mails address: Co.montasser.salman@uoanbar.edu.iq, Rabahalobaidy@uoanbar.edu.iq

ORCID:

1 https://orcid.org/0000-0001-9159-573X,

2 https://orcid.org/0000-0002-8347-2729,

3 https://orcid.org/0000-0002-5713-6916

Abstract:

Blockchain technology has the potential to revolutionize various industries by providing a secure and decentralized way of storing and verifying data. However, measuring the performance and scalability of Blockchain systems are often limited and required a simulator. In this paper, we propose the use of genetic algorithms for optimizing Blockchain simulator. Blockchain simulator, such as BlockSim, demonstrate the effectiveness of a series of experiments on different situations. Hence, an optimization algorithm is required. Our results show that Genetic algorithm can significantly optimize the performance and scalability of Blockchain systems.

Keywords: ***Blockchain, Genetic Algorithm, Optimization, Fitness Function, BlockSim.***

Introduction:

Blockchain technology has gained significant attention in recent years due to its potential to revolutionize various industries, including supply chain management, cloud computing, wireless sensor networks, and the Internet of Things (1). However, measuring the performance of Blockchain systems remains a challenge due to the complexity of the technology and excessively variables involved (2). One solution to overcome this issue is to use a simulator. BlockSim is a widely used simulator for modelling and analysing Blockchain systems in various fields, including manufacturing, logistics, and supply chain management (3).

Optimizing the performance and effectiveness of Blockchain simulator is a critical challenge, given the often limited resources and constraints that exist in real-world systems. This can involve optimizing the scheduling of operations, the use of resources, and the overall performance of the system, among other aspects. Genetic algorithms are a powerful optimization technique that can be applied to a wide range of problems, including those encountered in Blockchain systems (4). Based on the principles of natural selection and evolution, genetic algorithms use a population of potential solutions (called chromosomes or

genomes) that are iteratively improved through the application of genetic operators such as crossover, mutation, and selection (5). By exploring a large search space and adapting to changing conditions, genetic algorithms can find good solutions to problems that may be difficult to solve using other methods (1). In recent years, genetic algorithms have been applied to various optimization problems in Blockchain systems and other simulation systems, with promising results (6–8).

In this paper, we examine the use of genetic algorithms for optimizing various aspects of BlockSim. First, we present a general framework for applying genetic algorithms to BlockSim optimization. Then we describe several specific approaches that can be used in different scenarios. We also discuss the challenges and limitations of using genetic algorithms for BlockSim optimization and suggest directions for future research. Our results demonstrate the effectiveness and efficiency of using genetic algorithms for optimizing BlockSim, and highlight the potential benefits of this approach for a wide range of applications.

The remainder of this paper is organized as follows: In the next section, we provide a review of the background researches on Blockchain simulator optimization (more specifically BlockSim) and genetic algorithms, including their characteristics and applications. Then, we present several examples of how genetic algorithms have been used to optimize BlockSim, including specific problems that were addressed, methods and approaches that were used, and results that were achieved. In results and analysis section, we present the findings of our optimization efforts and discuss the implications of our results. Finally, in conclusion, we summarize the main findings of the paper and suggest directions for future research.

Related Works

In this literature review, we explore the use of genetic algorithms as a tool for optimizing the performance of Blockchain systems. Several approaches had been done in this field. For example, (9) presents a method for optimizing Blockchain algorithms using genetic algorithms. This approach has the potential to improve the efficiency and scalability of Blockchain systems (10) focuses on optimizing Blockchain algorithms, specifically for use in the Internet of Things domain. The authors propose the use of EC-El Gamal encryption in conjunction with genetic algorithms to achieve a lightweight and scalable Blockchain solution.(11) presents a new problem formulation for optimizing the energy consumption of Blockchain-based systems using evolutionary algorithms. This work is relevant as energy consumption is a major concern in the operation of Blockchain systems.(12) proposes a mathematical method for optimizing hybrid consensus algorithms in Blockchain systems. The authors demonstrate the effectiveness of their method through its application in a Blockchain system. (2) uses particle swarm optimization to analyse the earnings forecast of Blockchain based financial products. This work is relevant as it shows the potential for using optimization techniques in the analysis and prediction of the performance of Blockchain-based financial products.

All, of these papers demonstrate the potential for using optimization techniques, such as genetic algorithms and evolutionary algorithms, to improve the efficiency and scalability of Blockchain systems. They also highlight the importance of addressing issues such as energy consumption in the design and operation of Blockchain systems.

Blockchain System

Blockchain systems are distributed ledger technologies that allow participants to reach consensus on the contents of the ledger without fully trusting each other. In recent years, there has been a surge of interest in Blockchain technology from academia, government, and industry. However, evaluating the design and implementation of different Blockchain systems can be challenging due to the lack of tools and resources. In this section, we will discuss some of the approaches that have been proposed to analyse and optimize the performance of Blockchain simulators and more specifically BlockSim.

BlockSim Simulator

BlockSim is a discrete-event simulator designed to evaluate the performance of different Blockchain implementations. It was proposed by (13) as a flexible and extensible framework for building simulation models of Blockchain systems. BlockSim allows users to rapidly model and simulate various Blockchain systems by extending the existing models. It has been applied to simulate both Bitcoin and Ethereum, allowing researchers to investigate the impact of different conditions and configurations on the performance of these systems.

A. **Scalability**

Scalability is a key challenge for Blockchain systems, as they must be able to handle an increasing number of transactions and users without sacrificing security or performance (14).

There are several approaches that can be taken to improve scalability in Blockchain systems:

- Sharding: This involves dividing the Blockchain into smaller pieces, or "shards," which can be processed in parallel. This can allow the system to handle more transactions simultaneously (15).
- Off-chain transactions: Allowing some transactions to be completed off the Blockchain can help reduce the number of transactions that need to be processed on-chain, improving scalability (16).
- Layer 2 solutions: These are protocols that build on top of existing Blockchains and allow for additional transactions to be processed outside of the main Blockchain. Examples include the Lightning Network for Bitcoin and Plasma for Ethereum (17).
- Increasing block sizes: Allowing blocks to hold more transactions can increase the capacity of the Blockchain to handle more transactions (18).
- Improved consensus algorithms: Some consensus algorithms, such as Proof of Stake, are more scalable than others, such as Proof of Work.
- Optimizing code: As with any system, optimizing code and improving the efficiency of the Blockchain can help improve scalability.
- Interoperability: Allowing different Blockchains to communicate and interoperate can help distribute the load across multiple systems, improving scalability (18).

B. **Network**

The network architecture and communication protocols used in a Blockchain system can significantly impact its performance. For example, the use of peer-to-peer networks and encryption can affect the delay in propagating blocks (13). Researchers have proposed various network-based approaches for improving the performance of Blockchain systems, such as the use of different topologies (19) and the optimization of communication protocols (20).

C-Parameters

In the BlockSim simulator, there are several key parameters that can impact the performance of a Blockchain system. These parameters include the block interval, block size, block propagation delay, block reward, transaction modelling technique, transaction propagation delay, transaction fee, transaction size, and the number of nodes in the system. Additionally, the number of simulation runs can also be configured. To study the impact of these parameters on the scalability and security of the Blockchain system researchers have used BlockSim to analyse the effects of different parameter settings on the performance of Blockchain systems .such as the decentralization of the Ethereum mining process in the simulation study of uncle blocks rewards (13). The block interval parameter determines the average time it takes to generate a new block, while the block size parameter specifies the size of each block in megabytes. The block delay parameter represents the propagation delay of blocks, and the block reward parameter determines the reward for generating a new block. The transaction modelling technique parameter controls the method used to model transactions, with options for full or light modelling. The transaction delay parameter specifies the propagation delay of transactions, while the transaction fee parameter determines the fee for each transaction. The transaction size parameter specifies the size of each transaction in megabytes. The number of nodes in the system parameter specifies the total number of nodes in the system, and the number of simulation runs parameter determines the number of simulation runs to be performed. By configuring these parameters, researchers can study the effects of different block and transaction characteristics on the performance of a Blockchain system.

Optimization Techniques

Optimization Techniques are a class of algorithms that aim to find the optimal solution to a problem by minimizing or maximizing a certain objective function. These algorithms are widely used in various fields such as computer science, engineering, and economics.

Optimization Techniques can be classified into several categories based on the nature of the optimization problem. Some common categories include linear programming, nonlinear programming, integer programming, and combinatorial optimization. Each category has its own set of optimization algorithms that are suitable for solving problems within that category.

Optimization algorithms can also be classified based on the way they search for the optimal solution. Some common search strategies include gradient descent, hill climbing, and simulated annealing. These algorithms start with an initial solution and iteratively improve it until the optimal solution is found.

Genetic Algorithms

Genetic algorithms are a class of optimization algorithms that are inspired by the principles of natural evolution. These algorithms are based on the idea of survival of the fittest and use the concept of selection, crossover, and mutation to search for the optimal solution.

In a genetic algorithm, a population of solutions (also called chromosomes) is initialized randomly. Each solution is evaluated based on its fitness, which is a measure of how well it solves the optimization problem. The solutions with higher fitness are more likely to be selected for reproduction.

In the reproduction stage, the solutions are combined through a process called crossover, which creates new solutions by combining the attributes of two parent solutions. The new solutions

are then subjected to a process called a mutation, which introduces random changes to the solutions.

The process of selection, crossover, and mutation is repeated until the optimal solution is found or a certain number of iterations is reached. Genetic algorithms are useful for solving optimization problems that are too complex to be solved by traditional optimization algorithms.

In Fig 1, the genetic algorithm is a heuristic optimization method that is inspired by the process of natural evolution. It begins by initializing a population of individuals, which represent potential solutions to the problem at hand. Each individual is evaluated using an evaluation function, which assigns a fitness score based on how well the individual solves the problem. The genetic algorithm then applies a set of genetic operators - selection, crossover, and mutation - to the current population to produce a new generation of individuals. This process is repeated until a stopping criterion is met, at which point the best individual found is returned as the solution. The genetic algorithm is known for its ability to search large solution spaces effectively, making it a popular choice for tackling complex optimization problems.

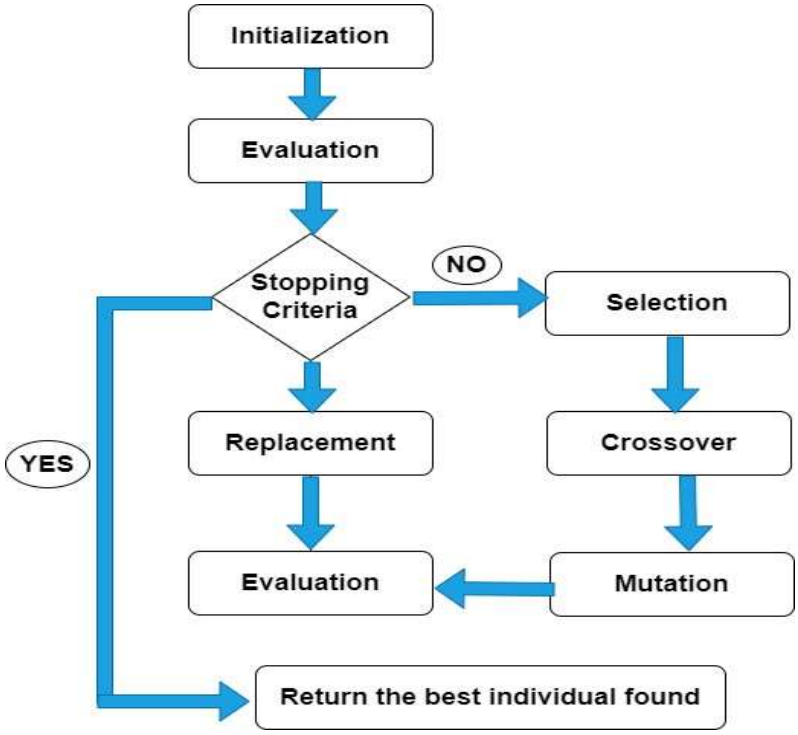

Figure 1 Genetic algorithm

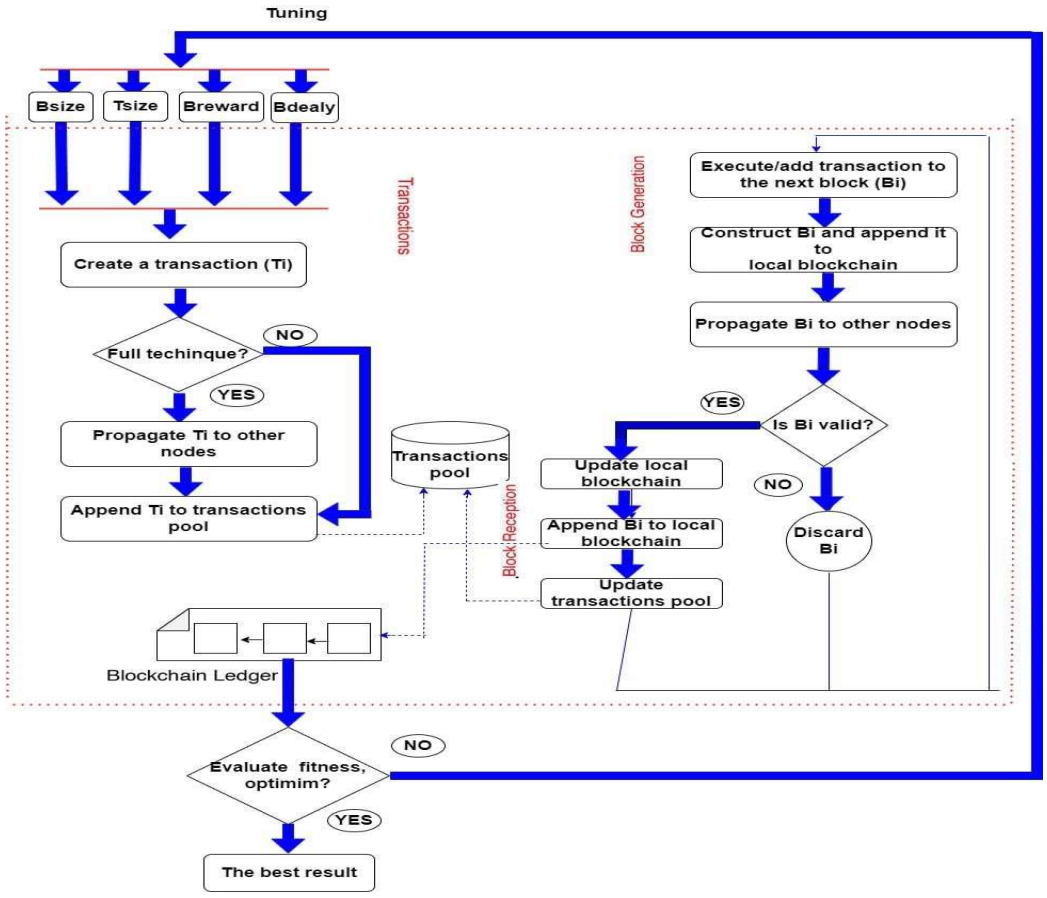

Figure 2 Flow chart of our optimization

Pseudo-code of Genetic Algorithm described in the following (21):

- Generate an initial population of individuals (chromosomes). These individuals represent potential solutions to the problem being solved.
- Evaluate the fitness value of each chromosome in the population. The fitness value is a measure of how well the chromosome solves the problem.
- Perform selection on the population. This involves selecting the fittest individuals from the population to serve as parents for the next generation.
- Perform crossover and mutation operations on the parents to create offspring. Crossover involves combining the genetic material of the parents to create new chromosomes, while mutation involves making random changes to the genetic material of the chromosomes.
- Perform a local search on the offspring. This involves making small, local changes to the offspring in an attempt to further improve their fitness.
- Repeat steps above until the genetic algorithm has run for the desired number of iterations or the stopping criteria have been met.

Select the best chromosome from the final population as a solution to the problem. This is the chromosome with the highest fitness value (21).

Methods

GA is a potent optimization method that can be utilized in situations when the link between the objective function and the choice variables is too complex to be expressed as it is in the BlockSim optimization. The first step in the GA process is to define the objective function, which will evaluate the quality of each solution. The objective function calculated using either a look-up table or computer simulation. The goal is either to maximize or minimize the objective function.

Once the objective function is defined, a group of encoded solutions is generated as an initial population. This population undergoes an iterative cycle of assessment and replication. During the evaluation step, every chromosome is assigned a fitness value according to how well it solves the objective function. This fitness value represents the chromosome's effectiveness in solving the problem.

The three fundamental reproduction operators of selection, crossover, and mutation are then used to carry out reproduction. Each chromosome is created using a varied number of copies by the selection operator, with more copies going to the better solutions. To make new chromosomes, a crossover operator swaps gene segments between chromosomes. The chromosome's individual alleles are changed at random by the mutation operator.

The evaluation and reproduction cycle repeats until the population reaches an optimal outcome or until other terminating requirements are met. The simulation is performed using ***mealpy Library*** (21) and the results show an increase in fitness cost over the number of generations. The successful application of GA in Blockchain systems tuning suggests that GAs are useful for balancing the profits of the BlockSim system.

The genetic algorithm is especially helpful for issues when the connection between the fitness function and the choice factors is ambiguous or complex. Thus, the procedures necessary to optimize a GA for Blockchain can be summed up as follows:

Representation of solutions: The problem is translated into a set of variables, called chromosomes, which represent potential solutions.

Objective Function: For the purpose of assessing the fitness of each solution, an objective function is defined. Either maximization or minimization is the objective function's purpose, which transfers the variables to a scalar value.

Initial Population: A starting population of coded solutions is produced at random or with the use of prior knowledge.

Evaluation: According to the objective function, a fitness value is given to each chromosome.

Reproduction: Applying reproduction operators like selection, crossover, and mutation to the previous generation results in the creation of the chromosomes of the new generation.

Selection: During further operations, the selection operator makes various numbers of copies of each chromosome. More copies are given to better solutions, following the "survival of the best" rule.

Crossover: In order to generate fresh results, the crossover operator switches individual genes between chromosomes. One-point crossover, multi-point crossover, and uniform crossover are only a few examples of the various crossover techniques that can be used.

Mutation: The mutation operator has a very low probability of changing individual alleles at random sites on random chromosomes.

Termination: Up until the population reaches an ideal solution or until additional termination requirements are met, evaluation and reproduction are iterated.

The successful application of GA in BlockChain optimization suggests its usefulness in balancing the profits of the BlockSim system. The simulation is performed using a GA package, such as ***mealpy Library***, and the parameters of the GA with BlockSim can be adjusted based on the specific requirements of the problem. The fitness cost of the best population in each generation is recorded and plotted to understand the evolution of the population over time.

Results and analysis

The fundamental contribution of this study is the proposal of GA as an optimization technique for the Blockchian simulator (i.e. BlockSim) to investigate the inherent trade-offs in parameter difficulty modification. Four key characteristics that have a significant impact on how well the Blockchian performs are identified through simulations, along with their applicability to the GA optimization. The simulator difficulty adjustment mechanism will still function in addition to the suggested way. In actuality, it fulfils the same function, which is to control the benefit level at which a simulator is provided. In order to maximize the fitness function of BlockSim required to mine a block, the GA assists by selecting the optimum settings to utilize. To ensure consistency across all trials, the deterministic halting condition in the GA is set to the entire number of epoch (i.e., actual time) required to mine the blocks. If not, trials may yield different results and the state trade-off would not be preserved. Four parameters from BockSim simulator are taken into account; block interval, block size, block propagation delay, and block reward. A minimum and maximum value of each parameter are listed in table (1).

Table (1) Selected parameters from BockSim simulator

No.	Parameter	Min.	Max.
1	Block interval	256	2048
2	Block size	0.1	2.0
3	Block propagation delay	0.5	16
4	Block reward	0.2	3

The block propagation latency and block reward might both be made as low as possible in this study. We used the multi-objective GA, because there are at least two optimization variables to take into account.

A modified version of BlockSim (13), was used to run the simulations. The default values for the other parameters during the simulation are listed in table (2).

Table (2) Simulation parameters for BockSim simulator

No.	Parameter	Description	Value
1	Tdelay	Propagation delay of transactions in seconds	5.1
2	Tn	Rate at which transactions can be created	10
3	Tfee	Transaction fee	0.000062
4	Nn	Total number of nodes in the network	3
5	Ttechnique	Technique for modeling transactions	Full/Light
6	Tsize	Transaction size in MB	0.000546

7	Bsize	Block size in Megabyte (MB)	30.0
8	Bdelay	Propagation delay of blocks in seconds	0.42
9	Breward	Block generation reward	12.5
10	Binterval	Average time to generate a block in seconds	600
11	hasTrans	Enable/Disabled transactions	True/False
12	simTime	Length of the simulation time	10000
13	Runs	Number of simulation runs	2

The GA was configured to run continuously during the simulation in order to determine whether the fitness value for mining N blocks increased in comparison to the previous one. Each set of optimization variables was used to mine 10,000 blocks, and it provided an approximation of the mining state for each set.

According to previous section, the following figure shows results of implementing GA for BlockSim simulator. As it can be seen, all curves had been done for 500 epoch. This required a lot of time (more than 7 days) on a Laptop with Core i7 and RAM 8 GB. Nine different cases had been studied and analysed in this paper. Starting with crossover probability 80, 85, and 90%; three different mutation probability 2, 5, and 7% had been tested respectively.

As shown in figure [3] (a,b and c) with C=80% and M=2,5,and 7 ; the fitness value become stable and constant after 140 Epoch with 104.9, 111.5 and 103.5 respectively . In the same way in figure [3] (d,e and f) with C=85% and M=2,5,and 7 ; the fitness value become stable and constant after 390 Epoch with 109.5, 107.7 and 100.8 respectively. While in figure [3] (g, h and I) with C=90% and M=2,5,and 7 ; the fitness value become stable and constant after 402 epoch with 106.7, 116.8 and 101.9. This indicate that changing the parameters of GA (i.e Crossover and Mutation) does not affect the fitness values greatly while number of Epoch increase rapidly (i, e. execution time consuming).

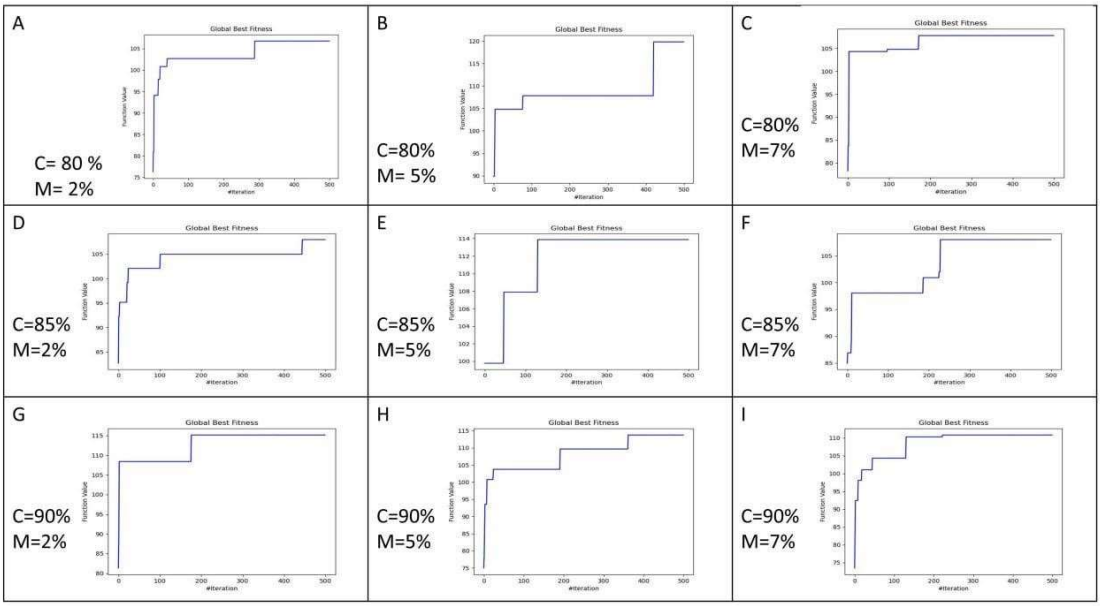

Figure 3: Implementing GA for BlockSim simulator with different crossover (c) and mutation (M) percent

On the other hand, the optimized Blocksim parameters are listed in table (3) shown below.
Table (3) Optimized parameters for BockSim simulator with GA

No	Crossover	Mutation	Tsize (sec.)	Bsize (Byte)	Bdelay(sec.)	Breward ($)	Fitness function
1	80	2	322.82270073	1.34105189	12.65671376	2.99876566	104.975620792664
2	80	5	1.78145050	1.10569741	9.05736687	2.93491957	111.534608977934
3	80	7	1242.6770578	1.485292	12.69206444	2.95711455	111.534608977934
4	85	2	1157.55717719	1.93999859	6.992562 02	2.95925349	109.504212657994
5	85	5	1.13023989	1.58610517	3.89795446	2.99215840	107.718591493484
6	85	7	6.32613456	4.78420530	2.03641312	2.96678096	109.504212657994
7	90	2	709.77158233	0.98332712	13.69460935	2.96508865	106.752004619434
8	90	5	4.40371483	1.46687983	1.43410758	2.99729735	116.89887823429
9	90	7	1.11192758	2.50844800	1.20138084	2.99792002	101.931304006191

Conclusions and future works

In conclusion, genetic algorithms represent a powerful tool for optimizing the performance of Blockchain systems and offer promising opportunities for further research and development in this area. High powerful computer required for more investigation since this method is time consuming in general. On the other hand, more parameters effect in Blockchain system are possible to be optimized using GA if direct (or indirect) relation with profit is determined.
Future research could focus on exploring the potential applications of genetic algorithms for optimizing other aspects of Blockchain systems, as well as addressing the challenges and limitations identified in the existing literature. It would also be valuable to investigate the scalability and robustness of genetic algorithm-based approaches for optimizing Blockchain systems and to explore their potential for integration with other optimization techniques.

Authors' contribution

A paper is made between five types of contributions: Douaa H. Khalaf Conceived and designed the analysis; she also collected the data; Muntaser A. Salman contributed data or analysis tools; Rabah N. Farhan Performed the analysis. All authors participated in writing the paper.

References

1. Khan PW, Byun Y-C, Park N. IoT-Blockchain Enabled Optimized Provenance System for Food Industry 4.0 Using Advanced Deep Learning. Sensors [Internet]. 2020 May 25;20(10):2990. Available from: https://www.mdpi.com/1424-8220/20/10/2990. https://doi.org/10.3390/s20102990
2. Ning Z, Sun S, Wang X, Guo L, Wang G, Gao X, et al. Intelligent resource allocation in mobile blockchain for privacy and security transactions: a deep reinforcement learning based approach. Sci China Inf Sci [Internet]. 2021 Jun 25;64(6):162303. Available from: https://link.springer.com/10.1007/s11432-020-3125-y. https://doi.org/10.1007/s11432-020-3125-y
3. Tsai C-W, Chen Y-P, Tang T-C, Luo Y-C. An efficient parallel machine learning-based blockchain framework. ICT Express [Internet]. 2021 Sep;7(3):300–7. Available from: https://linkinghub.elsevier.com/retrieve/pii/S240595952100103X. https://doi.org/10.1016/j.icte.2021.08.014
4. Kheiri F. A review on optimization methods applied in energy-efficient building geometry and envelope design. Renew Sustain Energy Rev [Internet]. 2018 Sep;92:897–920. Available from: https://linkinghub.elsevier.com/retrieve/pii/S1364032118302624
5. Hasbach JD, Witte TEF. Human-Machine Intelligence: Frigates are Intelligent Organisms. In: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC) [Internet]. IEEE; 2021. p. 1495–500. Available from: https://ieeexplore.ieee.org/document/9658640/. https://doi.org/10.1016/j.rser.2018.04.080
6. Li H, Weng S, Tong J, He T, Chen W, Sun M, et al. Composition of Resource-Service Chain Based on Evolutionary Algorithm in Distributed Cloud Manufacturing Systems. IEEE Access [Internet]. 2020;8:19911–20. Available from: https://ieeexplore.ieee.org/document/8968322/, doi: 10.1109/ACCESS.2020.2969234
7. Paprocka I, Kalinowski K, Balon B. The Concept of Genetic Algorithm Application for Scheduling Operations with Multi-resource Requirements. In 2021. p. 342–51. Available from: http://link.springer.com/10.1007/978-3-030-57802-2_33, dOI: 10.1007/978-3-030-57802-2_33
8. Vali-Siar MM, Gholami S, Ramezanian R. Multi-period and multi-resource operating room scheduling under uncertainty: A case study. Comput Ind Eng [Internet]. 2018 Dec;126:549–68. Available from: https://linkinghub.elsevier.com/retrieve/pii/S0360835218304844, https://doi.org/10.1016/j.cie.2018.10.014
9. Khan AA, Laghari AA, Gadekallu TR, Shaikh ZA, Javed AR, Rashid M, et al. A drone-based data management and optimization using metaheuristic algorithms and blockchain smart contracts in a secure fog environment. Comput Electr Eng [Internet]. 2022 Sep;102:108234. Available from: https://linkinghub.elsevier.com/retrieve/pii/S0045790622004700, https://doi.org/10.1016/j.compeleceng.2022.108234
10. Guruprakash J, Koppu S. EC-ElGamal and Genetic Algorithm-Based Enhancement for Lightweight Scalable Blockchain in IoT Domain. IEEE Access [Internet].

2020;8:141269–81. Available from: https://ieeexplore.ieee.org/document/9153770/, doi: 10.1109/ACCESS.2020.3013282

11. Alofi A, Bokhari MA, Bahsoon R, Hendley R. Optimizing the Energy Consumption of Blockchain-Based Systems Using Evolutionary Algorithms: A New Problem Formulation. IEEE Trans Sustain Comput [Internet]. 2022 Oct 1;7(4):910–22. Available from: https://ieeexplore.ieee.org/document/9739964/, doi: 10.1109/TSUSC.2022.3160491
12. Wu Y, Song P, Wang F. Hybrid Consensus Algorithm Optimization: A Mathematical Method Based on POS and PBFT and Its Application in Blockchain. Math Probl Eng [Internet]. 2020 Apr 13;2020:1–13. Available from: https://www.hindawi.com/journals/mpe/2020/7270624/, https://doi.org/10.1155/2020/7270624.
13. Alharby M, van Moorsel A. BlockSim: An Extensible Simulation Tool for Blockchain Systems. Front Blockchain [Internet]. 2020 Jun 9;3. Available from: https://www.frontiersin.org/article/10.3389/fbloc.2020.00028/full, https://doi.org/10.3389/fbloc.2020.00028.
14. Mazlan AA, Mohd Daud S, Mohd Sam S, Abas H, Abdul Rasid SZ, Yusof MF. Scalability Challenges in Healthcare Blockchain System—A Systematic Review. IEEE Access [Internet]. 2020;8:23663–73. Available from: https://ieeexplore.ieee.org/document/8968381/, doi: 10.1109/ACCESS.2020.2969230.
15. Dang H, Dinh TTA, Loghin D, Chang E-C, Lin Q, Ooi BC. Towards Scaling Blockchain Systems via Sharding. In: Proceedings of the 2019 International Conference on Management of Data [Internet]. New York, NY, USA: ACM; 2019. p. 123–40. Available from: https://dl.acm.org/doi/10.1145/3299869.3319889.
16. Cong K, Ren Z, Pouwelse J. A Blockchain Consensus Protocol With Horizontal Scalability. In: 2018 IFIP Networking Conference (IFIP Networking) and Workshops [Internet]. IEEE; 2018. p. 1–9. Available from: https://ieeexplore.ieee.org/document/8696555/
17. Zhou Q, Huang H, Zheng Z, Bian J. Solutions to Scalability of Blockchain: A Survey. IEEE Access [Internet]. 2020;8:16440–55. Available from: https://ieeexplore.ieee.org/document/8962150/, doi: 10.23919/IFIPNetworking.2018.8696555.
18. Kaur G, Gandhi C. Scalability in Blockchain: Challenges and Solutions. In: Handbook of Research on Blockchain Technology [Internet]. Elsevier; 2020. p. 373–406. Available from: https://linkinghub.elsevier.com/retrieve/pii/B9780128198162000150, https://doi.org/10.1016/B978-0-12-819816-2.00015-0.
19. Albshri A, Alzubaidi A, Awaji B, Solaiman E. Blockchain Simulators: A Systematic Mapping Study. In: 2022 IEEE International Conference on Services Computing (SCC) [Internet]. IEEE; 2022. p. 284–94. Available from: https://ieeexplore.ieee.org/document/9860110/, doi: 10.1109/SCC55611.2022.00049.
20. Ren Y, Liu Y, Ji S, Sangaiah AK, Wang J. Incentive Mechanism of Data Storage Based on Blockchain for Wireless Sensor Networks. Mob Inf Syst [Internet]. 2018 Aug 29;2018:1–10. Available from:

https://www.hindawi.com/journals/misy/2018/6874158/, https://doi.org/10.1155/2018/6874158.

21. Weise T. Global optimization algorithms-theory and application. Self-Published Thomas W. 2009;361.