Al-Salam Journal for Engineering and Technology
Journal Homepage: http://journal.alsalam.edu.iq/index.php/ajest
e-ISSN: 2790-4822p-ISSN: 2958-0862

# The Genetic Algorithm Implementation in Smart Contract for the Blockchain Technology

**Hamssah Hussein Abdul-Sada[1], Furkan Rabee[1]**

[1]University of Kufa/ Faculty of Computer Science and Mathematics, Al Najaf, 54001, Iraq

DOI: https://doi.org/10.55145/ajest.2023.02.02.005
Received December 2022; Accepted February 2023; Available online February 2023

**ABSTRACT:** A blockchain (BC) is merely a decentralized ledger that records each transaction that takes place inside the network. A block is a unit of data that contains encrypted information about every transaction in the network's history. Decentralized technology implementation would necessitate increased security, enforce accountability, and may speed up the transaction from the current hierarchical structure to a decentralized, cooperative chain of command. By producing fresh offspring of those animals who are identifiable by their identification tags, a blockchain is used to conserve the data of a special species of endangered animals. This article presents a decentralized application (SONR DAPPs) by implemented genetic algorithm to forecast a brand-new kind of those animals. The implementation did by solidity framework. Even that the solidity has disadvantages of memory limitation, but the result shows a possibility of using it to implement complex algorithms for the smart contract. This experience gives a chance to develop the solidity for a more complex algorithm to increase the ability o-f choosing better user to be a new block in the chain of this technology.

**Keywords:** Blockchain, Smart contract, Genetic algorithm, Endangered animals.

## 1. INTRODUCTION

Satoshi Nakamoto was the person who initially proposed the idea of blockchain [1] Lapis coins are the initial use case for blockchain. Blockchain platform Ethereum [2]. A peer-to-peer network's nodes share and maintain a distributed, chronological database of transactions called a blockchain platform [3]. A blockchain's decentralized design makes it possible to confirm transactions without the involvement of a central authority (e.g., a bank or a credit card company). The Ethereum platform is gradually garnering media attention, with pieces about it appearing in venerable publications like The Economist [4] and The New York Times [5] With the assistance of market leaders like Intel, Microsoft, J.P. Morgan, and Accenture, enterprise versions of the Ethereum platform are already being developed. Smart contracts are at the core of the Ethereum platform [6]. Simply described, a smart contract is a general-purpose computer program that cannot be altered. Ethereum, unlike app stores, not only holds smart contracts but also puts them to use. Solidity, which has a syntax akin to JavaScript, is the language used to write smart contracts. A tradeable digital token that may stand in for money, an asset, a virtual share, a membership proof, etc. is frequently created using smart contracts. Additionally, smart contracts frequently specify how these tokens are to be distributed.

Due to the development of blockchain technology, the phrase "smart contract," which was first used to describe the broad automation of legal contracts, has recently attracted a lot of attention. The phrase is now frequently used to describe scripts written in low-level code that execute on a blockchain network. Our research is limited on developing decentralized applications [7]. Blockchain applications are currently being used in various industries with smart contract development. In DApps apps, smart contracts are written using Solidity [8][9] . Solidity was first created to develop contracts for the Ethereum platform and incorporates object-oriented characteristics [10]. Programs written in Solidity may be tested and debugged on Remix and Visual Studio Code. VS code may be installed on a PC and used offline while using Remix in a web browser. An open-source application called the remix makes it possible to create Solidity contracts directly from a web browser. Writing and debugging Solidity contracts is made easier. Remix maintains usage both locally and, in the browser, because it is written in JavaScript. Remix is used to deploy, test,

*Corresponding author: hamssah.alhassnawe@student.uokufa.edu.iq and debug smart contracts [11]. Ethereum DApps and smart contracts are used together. Ethereum's mission is to provide a substitute protocol for building decentralized apps [12], Smart contracts and decentralized apps are becoming important topics [13], The use of DApps resulted in the generation of a lot of data. We select Ethereum DApps, the largest DApps market. We provide some background information on the section on Ethereum blockchain and Ethereum DApps. [14].

### 1.1 ETHERIUM BLOCKCHAIN

The blockchain is a decentralized ledger system that tracks value transfers between accounts. It is built on a P2P network. Every node in the network receives transactions, groups them into a block that is connected to the block before it, and broadcasts the block. The consensus method states that a block will become a part of the ledger if the majority of nodes receive and accept it. After Bitcoin, The second-generation blockchain is Ethereum. The cryptocurrency Ether is supported by the Ethereum ledger (ETH). User accounts and smart-contract accounts are the two types of accounts available on Ethereum.

**The user accounts represent** participants, including miners, deployers, and callers (who use the functionality of smart contracts), as well as (whose nodes work to do contribution to the ledger).

**The smart-contract accounts** include Chaincode (code on chain) is another term for the smart contract, a category of programs that may be stored on and executed on blockchains.

The first blockchain to offer a Turing-complete programming language for creating smart contracts is Ethereum. Transfers of ETH and the execution of contracts are two different types of transactions. An ETH transfer is the act of moving some ETH from one user account to another. An account calling a smart contract function with some data as the input and some ETH as the cost for contract execution is referred to as a contract execution. The transaction initiator, on the other hand, asserts that transactions may be divided into two categories: internal transactions, launched by smart contract accounts, and external transactions, started by user accounts. The link between the two classes is seen in (Figure 1). A transaction is considered an ETH transfer if the target account is a user account. If the destination account is a smart-contract account, a transaction is considered to be contract execution. All transactions need a charge from the account. In the Ethereum ecosystem, these charges are all referred to as gas abstract-text for the abstract text etc.

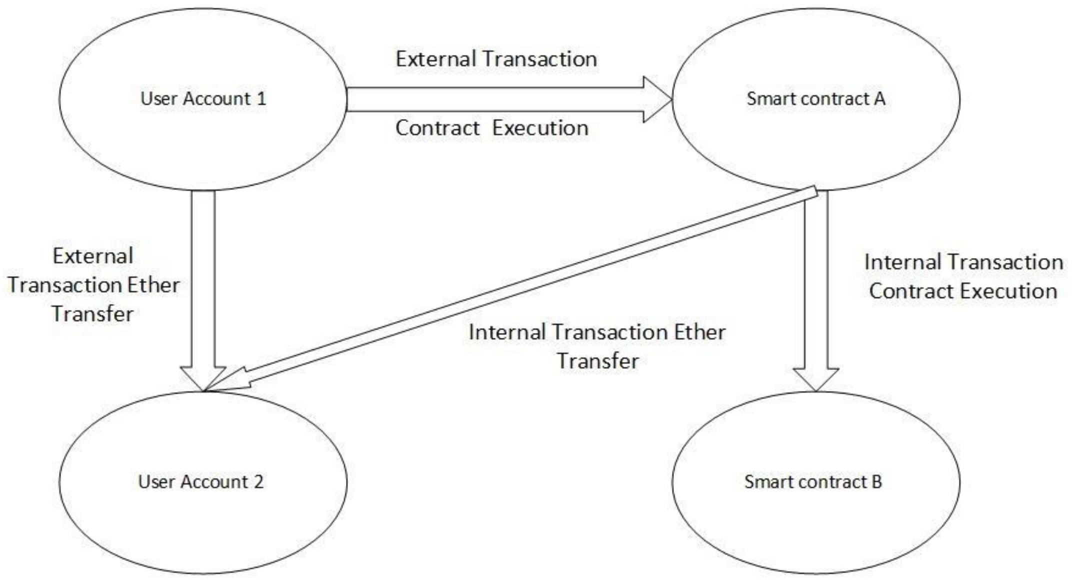
User Account 1
External Transaction
Contract Execution
Smart contract A
External Transaction Ether Transfer
Internal Transaction Contract Execution
Internal Transaction Ether Transfer
User Account 2
Smart contract B

FIGURE 1. **Relationship between the two types of transactions.**[14]

### 1.2 ETHEREUM DECENTRALIZED APPLICATION (DAPP)

Utilizing smart contracts, the Ethereum blockchain offers computing and storage capabilities. As a result, Ethereum DApps may employ smart contracts to execute business logic using the features offered by Ethereum. For complete decentralization, all the operations and data associated with A DApp built on blockchain should be managed and kept there. However, contemporary DApps often only implement a portion of their functionality on the blockchain due to the unique performance of blockchain systems. User interface (Front End) and server side are the two primary components of standard or traditional web applications that allow and make them useful (back end). HTTP protocols are used for communication between the front and back ends. Similar to applications that are often housed on centralized architectures, which raise the risk of a single point of failure, this sort of application has several issues. By assaulting the hosting provider, a malicious attack might disrupt data and seriously destroy it. However, the backend of DApps is similar to the Ethereum blockchain but the user experience (front end) is identical to that of regular apps. In DApps, the front end and back end interact and communicate in the same way as in regular applications, but the user cannot tell the difference. DApps make it very hard to shut down any application because doing so would require shutting down every distributed node. DApps might function on a blockchain or a peer-to-peer network [15].

As a result, Ethereum DApps in reality use one of three types of architectures: direct, indirect, or hybrid (Figure 2). In direct architecture DApps (Figure 2A), the client communicates directly with E'thereum-deployed smart contracts. The client communicates with the smart contract through the server in DApps that use the indirect architecture (Figure 2B), which has back-end services operating on a centralized server. DApps with a hybrid design combine the two architectures mentioned above, allowing for both direct and indirect interactions with smart contracts via a back-end server.

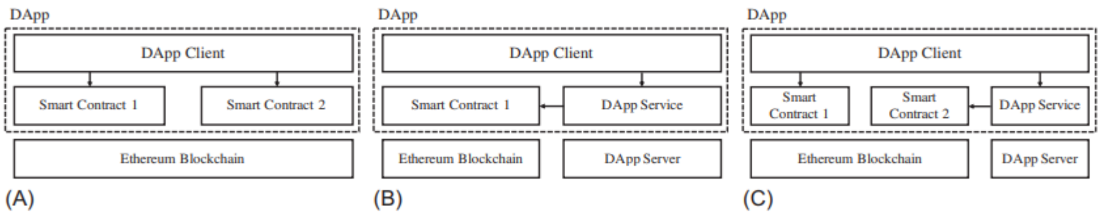

**FIGURE 2. Three kinds of decentralized application (DApp) architectures. A, Direct; B, Indirect; C, Mixed.**[14]

## 1.3 COST OF SMART CONTRACT

Deployment and contract execution are the two components that make up a smart contract's cost. A unique function that is in charge of deploying smart contracts is the constructor function. The contract's complexity may be reflected in the deployment cost. The amount gas transferred in transactions inside a block is capped due to the fact that executions are consistently encoded in transactions before being combined into blocks. Any contract execution that exceeds the limit in gas prices will fail and make no difference. the following scenarios' estimated costs for deploying smart contracts on Ethereum:

### 1.3.1 Ethereum Cost Charging - that may depend on the below-listed factors:

- Gas fees
- Formation of contract
- Storage under contract
- Execution of a contract

### 1.3.2 Smart Contract Cost Charging- that may depend on the following factors:

- How many individuals are involved in the creation of smart contracts?
- Tools and technology needed for Ethereum's adoption of smart contracts
- Cost of project management
- Post-delivery and maintenance services

## 1.4 SOLIDITY

The Ethereum community uses the computer language Solidity to create smart contracts. It is a language that resembles JavaScript and has contracts (similar to classes), functions, and events. A smart contract's source code is compiled into bytecode before being implemented on Ethereum. The smart contract will receive an address upon deployment, put smart contracts into use. Smart contracts can be used by all accounts. Sometimes programmers install additional child contracts using a smart contract.

## 1.5 GENETIC ALGORITHM IN BLOCKCHAIN SPACE

Developed by Holland in 1975[16], the genetic algorithm (GA) is a paradigm for group computing processes. This approach, which is driven by the chromosomal survival principle, looks for the best answer to a particular problem. With this approach, a population made up of strings is initially established. Individuals referred to by the identifier make up this demographic. Genes are the source of both chromosomes and human beings. The population is then exposed to the chosen fitness function, and fitness outcomes that are near to the ideal outcome are identified. Three steps make up this process: crossover, mutation, and selection. The parent person is chosen in the selection process based on the population's average fitness levels. A portion of the parent persons chosen via the selection process is replaced during the crossover procedure. In this manner, IDs with novel traits are not present in the general population that is produced [16][17].

### 1.6 LIMITATIONS

**Limited stack size** With a maximum size of 1024 items and a maximum item size of 256, the stack (i.e., the region in which all calculations are performed) might become troublesome for large scripts.

**Lack of tools/techniques** to check the code's accuracy. Developers might make use of a variety of technologies in the traditional software development process to assist them assure the quality of their code. Unlike smart contracts, which also have a finite amount of compilation elements and factors affecting memory size and stack. The ongoing growth of solidity languages is the cause of all these challenges.

## 2. RELATED WORK

This section discusses methods that are currently available on the Ethereum network for creating decentralized applications.

**CryptoKitties, Ethereum; (2017)** In the simulation game CryptoKitties, players breed, acquire, and trade virtual cats, which are truly one-of-a-kind ERC-721 tokens. The smart contracts' functionality includes cat-related businesses like breeding and selling. To improve player interaction with Ethereum, the game incorporates a web application interface. When one of the most well-known blockchain games first debuted [18].

**Ruhi (2019)** focuses on beginner-friendly architectural design for blockchain-oriented apps and a method for determining which components of local application architecture can benefit from using Ethereum, In this study, a sample application development environment and smart contract example are provided. A step-by-step analysis is also done.[12].

**F.Blum(2020 )** suggests a systematic method utilizing currently used architectural principles, utilizes the Meta-Transaction design pattern to illustrate the use of strategies, tactics, and design patterns. By defining potential Strategies, Tactics, and Design Patterns in the context of Blockchain applications as a basis for a cogent decision process, this study serves as a first step towards such recommendations[19].

**K.Wu(2021)** Based on a sizable dataset of 995 Ethereum DApps and 29,846,075 transaction logs over them, this article delivers the first thorough empirical research of blockchain-based DApps to date. They provide a descriptive study of the DApp market, list the ways in which DApps access the underlying blockchain through smart contracts, and investigate the DApp deployment and operation concerns that merit attention. All of the findings have beneficial ramifications for various DApp ecosystem participants, such as end users, DApp developers, and blockchain providers [14].

**A.Bogner(2016)** This research demonstrates a decentralized app (DAPP) for sharing common items that is built on an Ethereum blockchain smart contract. Users may register and rent devices under this agreement without the assistance of a Trusted Third Party (TTP), the revelation of any personal information, or an earlier subscription to the service. The essential elements of the system design are a web application, a local Ethereum client, and a smart contract housed on the blockchain [20]**.**

**W. Buchanan (2022)** In the current study, they take into account two scenarios, one that utilizes the first two characteristics and another that serves to illustrate the concept of structural or governance decentralization. One of the more well-known decentralized applications is CryptoKitties (Bowles 2017). Axiom Zen developed a game that lets users buy, sell, trade, breed, and collect digital cats. Both conventional and digital games frequently feature rare or unique tradeable things; however, in CryptoKitties , the digital items are stored on the blockchain. In this sense, the activities are assured and visible, but the application's ethos is not really decentralized. The DAO (decentralized autonomous organization) is another illustration of a DApp (Securities and Exchange Commission 2017). Voting-capable tradeable tokens served as the application's primary form of governance. In this sense, the decentralized Ethereum layer insured the application's mechanism, while the idea itself was created to promote authority decentralization[21]**.**

**T.Min (2022)** They created a number of datasets for this article, totaling more than 73.8 million transactions produced by 230,000 addresses. Hexadecimal addresses are converted into understandable program names, and after that, depending on the DApp categories, user behavior is examined. Additionally, they used an unsupervised clustering technique on the 230,000 addresses to separate participants from investors and examine their behavioural tendencies and susceptibility to blockchain markets, such as ETH pricing[22].

**Z.Wang (2021)** evaluate the corpus of research on Blockchain-based smart contracts from the time they were first developed (in July 2015) to the time this book was written (in July 2019), and they offered a taxonomy focused on the major security problems and potential mitigation techniques. They also talked about the security flaws in the contract program architecture and other ways that smart contracts may be targeted and used maliciously. To develop contract security, certain dynamic research topics are also offered [23]**.**

**N.BAYGIN(2022)** proposed that one of the most significant issues with mass customisation is collaboration and trust, and a blockchain-based platform has been proposed to address these issues. Additionally, the issue of supply chain and production process optimization in the manufacturing industry has been looked at in this study. Numerous aspects are involved in this process, which extends from the manufacturer to the customer. Consequently, this process' optimization is a particularly challenging challenge. To address this issue, a two-stage approach has been developed. By using blockchain to connect service providers and customers in the manufacturing industry, a trustworthy platform was first established. The most appropriate parties were chosen using a genetic algorithm in the second phase [24].
**Z.Wei(2020)** focuses on a taint analysis technique based on the EVM is proposed to reduce the invalid input, a dangerous operation database is designed to identify the dangerous input, and genetic algorithm is used to optimize the code coverage of the input, which together construct the fuzzing framework for smart contracts. He selects the stopping condition as the population size equals 50 people with the beginning population being an integer data type for the redundancy problem of input data of fuzzing in order to shorten the screening time and assure the population variety [25].

To present a simple comparison between the proposed method in the paper with **Z.Wei(2020),** string sequences are used as an initial population in solidity, ASCII code of each character is converted to a binary code which is then concatenated in alphabetical order as the encoding of the string**.**

## 3. PROPOSED MODEL

Rare species preservation from extinction is a goal that most scientists aim for these days. The proposed DAPPs achieves this using virtual application that simulates the work of nature reserve.

A specific number of endangered animals are reproduced in the Ethereum network using a virtual application. (SONR) simulation of nature reserved uses the genetic sequence as ID tag and applies genetic algorithms on the same sequence.

### 3.1 IMPLEMENTED CONTRACT

One contract (SONR) is mainly used with multiple functions included, e.g finding a new offspring from two parents through a single crossover (crossover function).
In addition to The necessity of having an administrator with restricted privileges for changing the code in smart contract.The terminologies of GA and SONR DAPPs are mapped in Table 2 are used interchangeably in the paper.

**Table 2.** Mapping of GA and SONR DAPPs

| GA Terminologies | SONR Terminologies |
|---|---|
| Population | Set of possible ID |
| Chromosomes | ID |
| Gene | Character |
| Offspring | Newly generated ID |
| Parents | Selected ID for crossover |

### 3.2 STRUCTURE OF (simOfNaturR)

simOfNaturR is composed of 6 main functions

1- input the IDs for a group of animals in a string form
2- string conversion based on the alphabetical sequence
3- calculate fitness for each animal in each generation
4- select parent with the highest fitness after arranging them in a descending order
5- apply single-point crossover for the ID of two parent solutions are swapped before and after a single point.
6- add the new offspring to the original population.

The figure 3 describes the process
First:- input 4 animal IDs (string form).
Second:- convert string form to integer vector based on alphabet sequence .
Third:- calculate fitness of each ID and arrange them in descending order .
Fourth:- select parents with the highest fitness and perform single-point crossover producing a new offspring added to the initial population.

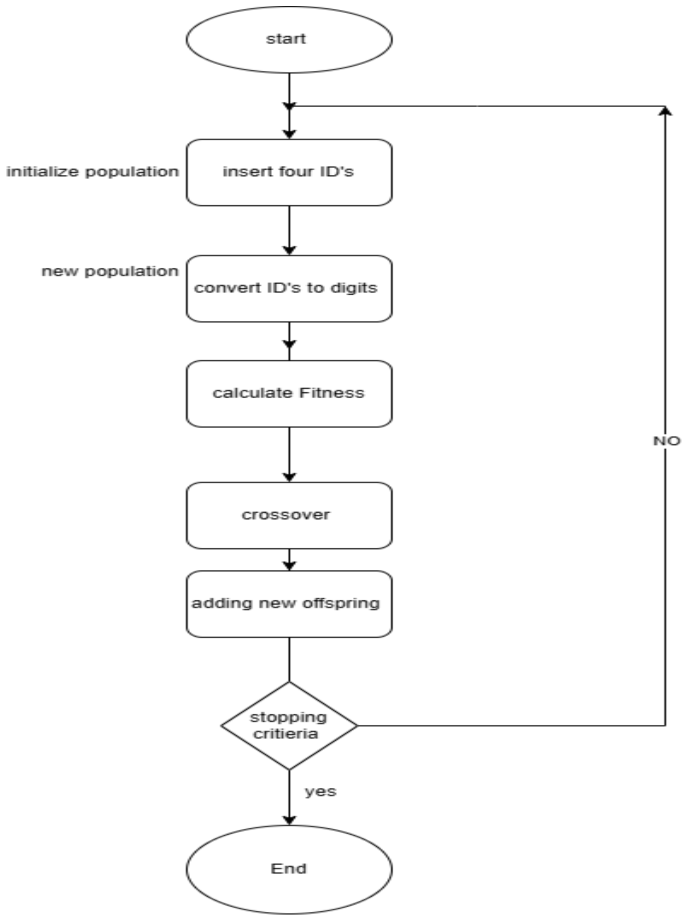

**FIGURE 3.** Proposed model (SONR)

## 3.3 simOfNaturR DEPLOYMENTS

The smart contract is written in Solidity language, then deployed on the blockchain network as shown in (figure 4).

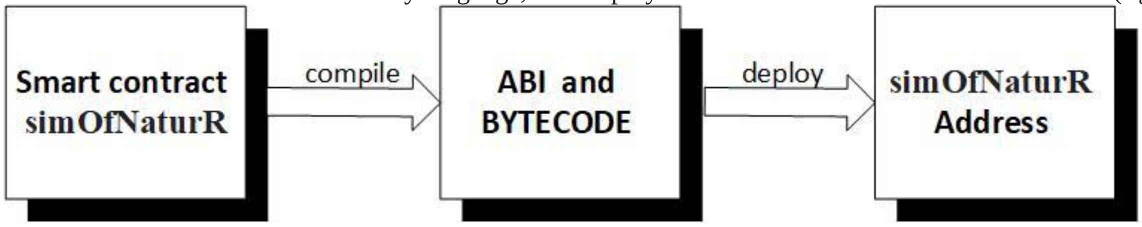

**Figure (4):** smart contract deployment

As shown in (Figure 5) the ID will enter a sequence of processes to find a new offspring of ID and soon .

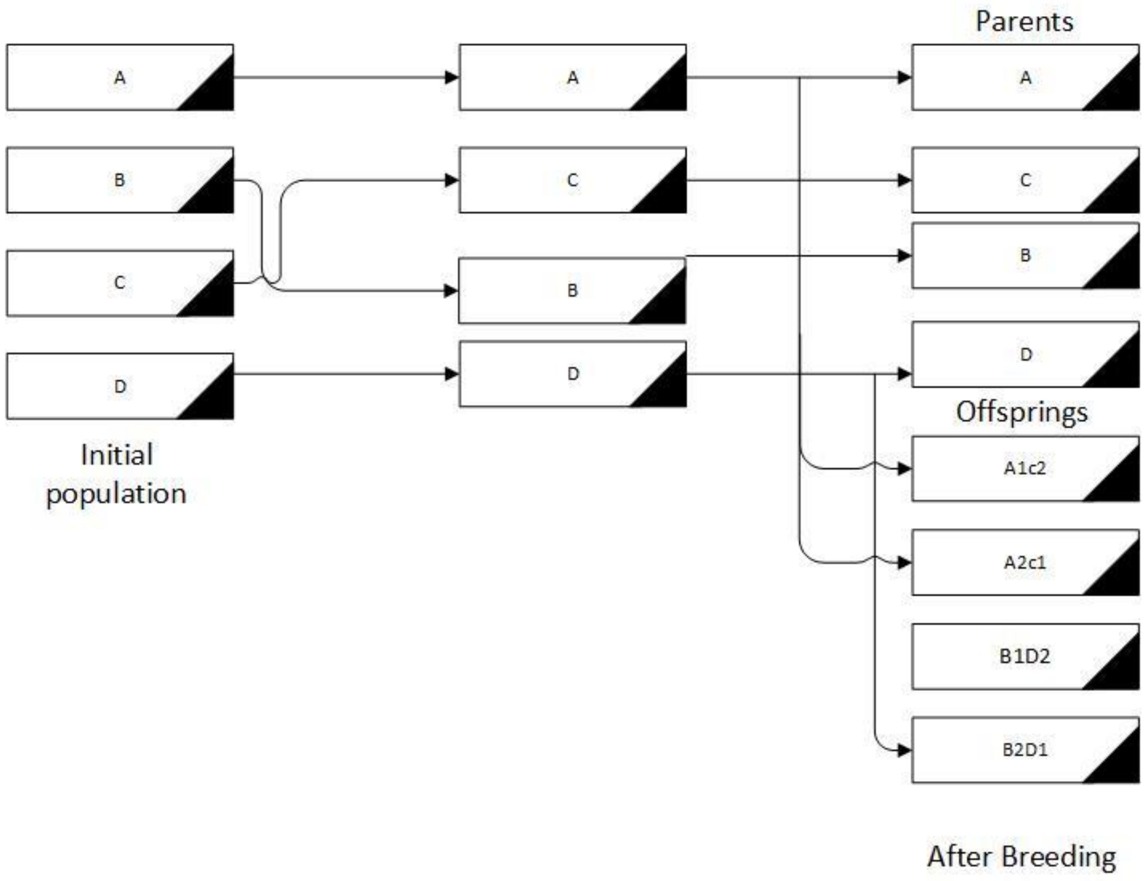

**FIGURE 4**. Steps of SONR DAPPs

## 4. EXPERIMENTAL RESULTS

Smart contract were created to implement the genetic algorithm in this study using solidity and Remix-IDE V0.29.0 .this learning environment has been studied using computer support processor AMD Ryzen 5 3500U with Radeon Vega Mobile GFX 2.10 GHZ ,12.0 GB RAM and 64-bit operating system ,X64 based processor. to enable the replication of our results, which are reported in Table 1. All of the experiments we took into consideration show that the finest solutions discovered by GAs can become more fit using the selection procedure. The starting population size, the number of IDs that are picked at each generation, the number of IDs that are created, and the probability of applying a crossover or a mutation during generations all affect the performance of all the GAs we have discussed. Five generation of genetic algorithm were computed and the parent ID for SONR was set to 18 character .It Calculate fitness function for all ID in each generation So that the process of selecting individuals is according to the descending order of fitness, then the crossover is applied, and the process is repeated until an optimal generation is obtained.

**Table 1** Results of GA for five generations

| Status | Generation # | Order | Fitness | String |
|---|---|---|---|---|
| 🟩 | **initial population** | **1** | **189** | **TTTGTTAATCAGCATCTT** |
| 🟩 | **initial population** | **4** | **98** | **AAACAATTAGTCGTAGAA** |
| 🟩 | **initial population** | **2** | **149** | **CAGGAGTTTGTGCGTGGC** |
| 🟩 | **initial population** | **3** | **106** | **TACAAAGCTTCAGGCCCT** |
| 🟩 | **second generation** | **2** | **167** | **TTTGTTAATGTGCGTGGC** |
| 🟩 | **second generation** | **4** | **146** | **CAGCATCTTCAGGAGTTT** |
| 🟥 | **second generation** | **7** | **98** | **AAACAATTATCAGGCCCT** |
| 🟩 | **second generation** | **5** | **106** | **GTCGTAGAATACAAAGCT** |
| 🟩 | **second generation** | **1** | **189** | **TTTGTTAATCAGCATCTT** |
| 🟥 | **second generation** | **8** | **98** | **AAACAATTAGTCGTAGAA** |

| | | | | |
|---|---|---|---|---|
| 🟩 | **second generation** | **3** | **149** | **CAGGAGTTTGTGCGTGGC** |
| 🟩 | **Third generation** | **1** | **192** | **TTTGTTAATGTGCGTGGC** |
| 🟩 | **Third generation** | **2** | **189** | **CAGCATCTTTTTGTTAAT** |
| 🟩 | **Third generation** | **5** | **154** | **CAGGAGTTTCAGGAGTTT** |
| 🟩 | **Third generation** | **8** | **141** | **GTGCGTGGCCAGCATCTT** |
| 🟩 | **Third generation** | **9** | **116** | **GTCGTAGAATCAGGCCCT** |
| 🟥 | **Third generation** | **14** | **96** | **TACAAAGCTTACAAAGCT** |
| 🟩 | **Third generation** | **4** | **167** | **TTTGTTAATGTGCGTGGC** |
| 🟩 | **Third generation** | **7** | **146** | **CAGCATCTTCAGGAGTTT** |
| 🟥 | **Third generation** | **12** | **98** | **AAACAATTATCAGGCCCT** |
| 🟩 | **Third generation** | **10** | **106** | **GTCGTAGAATACAAAGCT** |
| 🟩 | **Third generation** | **3** | **189** | **TTTGTTAATCAGCATCTT** |
| 🟥 | **Third generation** | **13** | **98** | **AAACAATTAGTCGTAGAA** |
| 🟩 | **Third generation** | **6** | **149** | **CAGGAGTTTGTGCGTGGC** |
| 🟥 | **Third generation** | **11** | **106** | **TACAAAGCTTCAGGCCCT** |

| **Status** | **Generation #** | **Order** | **Fitness** | **String** |
|---|---|---|---|---|
| 🟩 | **Fourth generation** | **1** | **240** | **TTTGTTAATTTTGTTAAT** |
| 🟩 | **Fourth generation** | **14** | **141** | **GTGCGTGGCCAGCATCTT** |
| 🟩 | **Fourth generation** | **2** | **192** | **TTTGTTAATGTGCGTGGC** |
| 🟩 | **Fourth generation** | **5** | **189** | **CAGCATCTTTTTGTTAAT** |
| 🟩 | **Fourth generation** | **10** | **149** | **CAGGAGTTTGTGCGTGGC** |
| 🟩 | **Fourth generation** | **8** | **154** | **CAGGAGTTTCAGGAGTTT** |
| 🟩 | **Fourth generation** | **16** | **138** | **CAGCATCTTCAGCATCTT** |
| 🟩 | **Fourth generation** | **11** | **149** | **CAGGAGTTTGTGCGTGGC** |
| 🟥 | **Fourth generation** | **19** | **106** | **GTCGTAGAAACAAAGCTT** |
| 🟥 | **Fourth generation** | **17** | **116** | **TCAGGCCCTGTCGTAGAA** |
| 🟩 | **Fourth generation** | **3** | **192** | **TTTGTTAATGTGCGTGGC** |
| 🟩 | **Fourth generation** | **6** | **189** | **CAGCATCTTTTTGTTAAT** |
| 🟩 | **Fourth generation** | **9** | **154** | **CAGGAGTTTCAGGAGTTT** |
| 🟩 | **Fourth generation** | **15** | **141** | **GTGCGTGGCCAGCATCTT** |
| 🟥 | **Fourth generation** | **18** | **116** | **GTCGTAGAATCAGGCCCT** |
| 🟩 | **Fourth generation** | **4** | **192** | **TTTGTTAATGTGCGTGGC** |
| 🟩 | **Fourth generation** | **13** | **146** | **CAGCATCTTCAGGAGTTT** |
| 🟥 | **Fourth generation** | **20** | **106** | **GTCGTAGAATACAAAGCT** |
| 🟩 | **Fourth generation** | **7** | **189** | **TTTGTTAATCAGCATCTT** |
| 🟩 | **Fourth generation** | **12** | **149** | **CAGGAGTTTGTGCGTGGC** |
| 🟩 | **Fifth generation** | **4** | **192** | **TTTGTTAATGTGCGTGGC** |
| 🟩 | **Fifth generation** | **1** | **240** | **TTTGTTAATTTTGTTAAT** |
| 🟩 | **Fifth generation** | **5** | **192** | **TTTGTTAATGTGCGTGGC** |
| 🟩 | **Fifth generation** | **6** | **192** | **GTGCGTGGCTTTGTTAAT** |
| 🟩 | **Fifth generation** | **10** | **189** | **CAGCATCTTTTTGTTAAT** |
| 🟩 | **Fifth generation** | **11** | **189** | **TTTGTTAATCAGCATCTT** |
| 🟩 | **Fifth generation** | **3** | **197** | **TTTGTTAATCAGGAGTTT** |
| 🟩 | **Fifth generation** | **25** | **146** | **CAGCATCTTCAGGAGTTT** |
| 🟩 | **Fifth generation** | **18** | **149** | **CAGGAGTTTGTGCGTGGC** |
| 🟩 | **Fifth generation** | **15** | **154** | **CAGGAGTTTCAGGAGTTT** |
| 🟩 | **Fifth generation** | **19** | **149** | **CAGGAGTTTGTGCGTGGC** |
| 🟩 | **Fifth generation** | **20** | **149** | **GTGCGTGGCCAGGAGTTT** |
| 🟥 | **Fifth generation** | **30** | **138** | **CAGCATCTTCAGCATCTT** |
| 🟩 | **Fifth generation** | **21** | **149** | **CAGGAGTTTGTGCGTGGC** |
| 🟥 | **Fifth generation** | **27** | **141** | **GTGCGTGGCCAGCATCTT** |
| 🟥 | **Fifth generation** | **31** | **138** | **CAGCATCTTCAGCATCTT** |
| 🟩 | **Fifth generation** | **2** | **240** | **TTTGTTAATTTTGTTAAT** |
| 🟥 | **Fifth generation** | **28** | **141** | **GTGCGTGGCCAGCATCTT** |
| 🟩 | **Fifth generation** | **7** | **192** | **TTTGTTAATGTGCGTGGC** |
| 🟩 | **Fifth generation** | **12** | **189** | **CAGCATCTTTTTGTTAAT** |

| | | | | |
|---|---|---|---|---|
| 🟩 | **Fifth generation** | **22** | **149** | **CAGGAGTTTGTGCGTGGC** |
| 🟩 | **Fifth generation** | **16** | **154** | **CAGGAGTTTCAGGAGTTT** |
| 🟥 | **Fifth generation** | **32** | **138** | **CAGCATCTTCAGCATCTT** |
| 🟩 | **Fifth generation** | **23** | **149** | **CAGGAGTTTGTGCGTGGC** |
| 🟩 | **Fifth generation** | **8** | **192** | **TTTGTTAATGTGCGTGGC** |
| 🟩 | **Fifth generation** | **13** | **189** | **CAGCATCTTTTTGTTAAT** |
| 🟩 | **Fifth generation** | **17** | **154** | **CAGGAGTTTCAGGAGTTT** |
| 🟥 | **Fifth generation** | **29** | **141** | **GTGCGTGGCCAGCATCTT** |
| 🟩 | **Fifth generation** | **9** | **192** | **TTTGTTAATGTGCGTGGC** |
| 🟩 | **Fifth generation** | **26** | **146** | **CAGCATCTTCAGGAGTTT** |
| 🟩 | **Fifth generation** | **14** | **189** | **TTTGTTAATCAGCATCTT** |
| 🟩 | **Fifth generation** | **24** | **149** | **CAGGAGTTTGTGCGTGGC** |

**FIGURE 5.** Shows the average fitness for each generation through a genetic algorithm. There is a progressive significant increase in the fitness value from the first to the fifth generation, which is the required individuals with the optimum genetic characteristics.

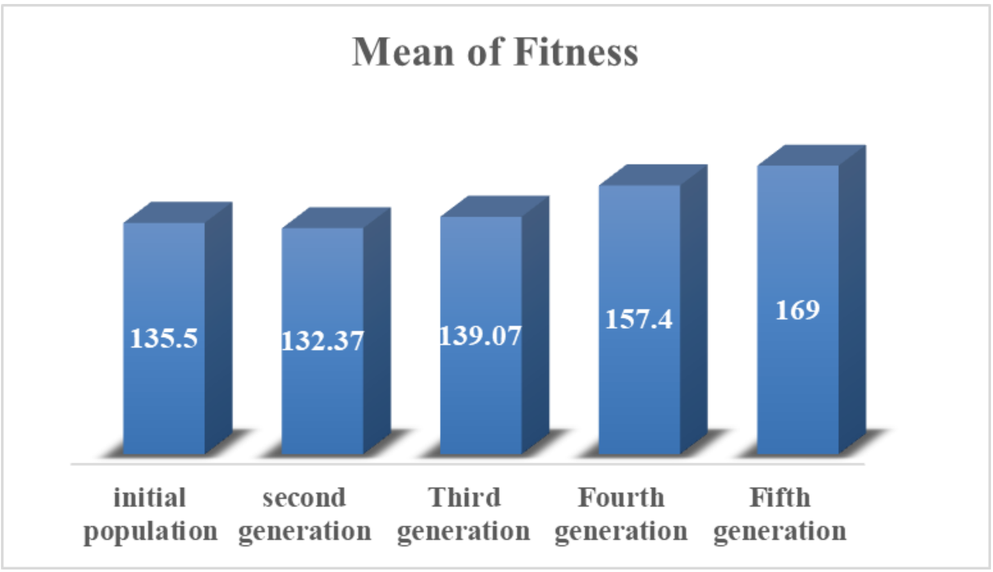

**FIGURE 7.** Shows the variance in fitness values throughout different generations according to the process of selecting IDs from the population with high fitness value and disregard IDs with low fitness values in order to obtain an optimum characteristic generation.

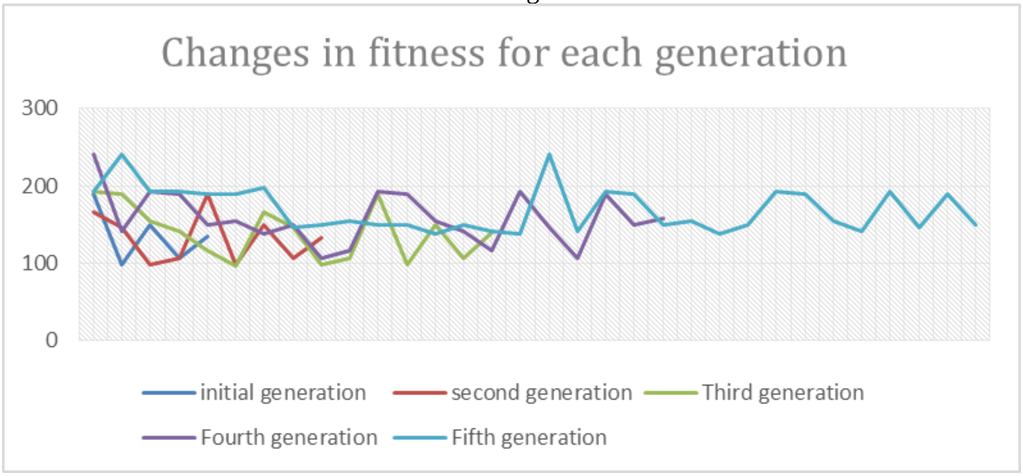

**FIGURE 8.** Variance of the fitness function for each generation

## 5. CONCLUSION AND FRAMEWORK

This study aims at selecting the best IDs out of the entire Blockchain population. GA is used here, which is one of the optimizing methods that use more than objective function, locate and find a solution to a difficult problem as protecting rare species from extinction.

In order to create a decentralized application that simulates a natural reserve and employs evolutionary algorithms to protect endangered species, this article introduces a new class of DAPPs called SONR. Future efforts should initially focus on improving the selection and mutation processes used to produce new progeny. Second, we'll investigate improved fitness metrics to assess the created ID. Finally, in order to overcome the environment's restriction on dynamic testing of smart contract programs, we will strive to address the restrictions of the smart contract execution environment. In accordance with generally acknowledged security standards, we seek to further enhance the produced application for scalability issues.

## Funding

None

## ACKNOWLEDGEMENT

I would like to thank the reviewers, editors and staff who contributed by reviewing my study.

## CONFLICTS OF INTEREST

The authors declare no conflict of interest

## REFERENCES

[1] S. Nakamoto and A. Bitcoin, “A peer-to-peer electronic cash system,” *Bitcoin.–URL https//bitcoin. org/bitcoin. pdf*, vol. 4, p. 2, 2008.
[2] G. Wood, “Ethereum: a secure decentralised generalised transaction ledger (2014).” 2017.
[3] M. Swan, *Blockchain: Blueprint for a new economy*. “ O’Reilly Media, Inc.,” 2015.
[4] M. Dhawan, “Analyzing safety of smart contracts,” in *Proceedings of the Conference: Network and Distributed System Security Symposium, San Diego, CA, USA*, 2017, pp. 16–17.
[5] N. Popper, “Understanding Ethereum, Bitcoin’s virtual cousin,” *New York Times*, vol. 1, p. 2017, 2017.
[6] N. Szabo, “Smart contracts http://www. fon. hum. uva. nl/rob/Courses/InformationInSpeech/CDROM/Literature,” *LOTwinterschool2006/szabo. best. vwh. net/smart. Contract. html Go to Ref. Artic.*, 1994.
[7] W. Zou *et al.*, “Smart Contract Development: Challenges and Opportunities,” *IEEE Trans. Softw. Eng.*, vol. 47, no. 10, pp. 2084–2106, 2021, doi: 10.1109/TSE.2019.2942301.
[8] Q. Xu, Z. He, Z. Li, and M. Xiao, “Building an ethereum-based decentralized smart home system,” in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, 2018, pp. 1004–1009.
[9] C. Dannen, *Introducing Ethereum and solidity*, vol. 1. Springer, 2017.
[10] P. Zhang, J. White, D. C. Schmidt, and G. Lenz, “Design of blockchain-based apps using familiar software patterns with a healthcare focus,” in *Proceedings of the 24th Conference on Pattern Languages of Programs*, 2017, pp. 1–14.
[11] “Remix - Ethereum IDE.” https://remix.ethereum.org/ (accessed Dec. 19, 2022).
[12] R. Taş and Ö. Ö. Tanrıöver, “Building a decentralized application on the Ethereum blockchain,” in *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2019, pp. 1–4.
[13] V. Buterin, “A next-generation smart contract and decentralized application platform,” *white Pap.*, vol. 3, no. 37, pp. 1–2, 2014.
[14] K. Wu, Y. Ma, G. Huang, and X. Liu, “A first look at blockchain-based decentralized applications,” *Softw. - Pract. Exp.*, vol. 51, no. 10, pp. 2033–2050, 2021, doi: 10.1002/spe.2751.
[15] J. Angelis and E. R. Da Silva, “Blockchain adoption: A value driver perspective,” *Bus. Horiz.*, vol. 62, no. 3, pp. 307–314, 2019.
[16] R. Z. Farahani and M. Elahipanah, “A genetic algorithm to optimize the total cost and service level for just-in-time distribution in a supply chain,” *Int. J. Prod. Econ.*, vol. 111, no. 2, pp. 229–243, 2008.
[17] S. Katoch, S. S. Chauhan, and V. Kumar, “A review on genetic algorithm: past, present, and future,” *Multimed. Tools Appl.*, vol. 80, no. 5, pp. 8091–8126, 2021.
[18] “CryptoKitties | Collect and breed digital cats!” https://www.cryptokitties.co/ (accessed Dec. 19, 2022).
[19] F. Blum, B. Severin, M. Hettmer, P. Huckinghaus, and V. Gruhn, “Building Hybrid DApps using Blockchain Tactics -The Meta-Transaction Example,” *IEEE Int. Conf. Blockchain Cryptocurrency, ICBC 2020*, 2020, doi: 10.1109/ICBC48266.2020.9169423.
[20] A. Bogner, M. Chanson, and A. Meeuw, “A decentralised sharing app running a smart contract on the ethereum blockchain,” *ACM Int. Conf. Proceeding Ser.*, vol. 07-09-Nove, pp. 177–178, 2016, doi: 10.1145/2991561.2998465.

[21] W. J. Buchanan, *Blockchain and Crypto-currency*. 2022. doi: 10.1201/9781003337751-11.

[22] T. Min and W. Cai, “Portrait of decentralized application users: an overview based on large-scale Ethereum data,” *CCF Trans. Pervasive Comput. Interact.*, vol. 4, no. 2, pp. 124–141, 2022, doi: 10.1007/s42486-022-00094-6.

[23] Z. Wang, H. Jin, W. Dai, K. K. R. Choo, and D. Zou, “Ethereum smart contract security research: survey and future research opportunities,” *Front. Comput. Sci.*, vol. 15, no. 2, 2021, doi: 10.1007/s11704-020-9284-9.

[24] “Genetic Algorithm-Based Optimization of Mass Customization Using Hyperledger Fabric Blockchain,” vol. 17, no. 2, pp. 451–460, 2022.

[25] Z. Wei, J. Wang, X. Shen, and Q. Luo, “Smart Contract Fuzzing Based on Taint Analysis and Genetic Algorithms,” *J. Quantum Comput.*, vol. 2, no. 1, pp. 11–24, 2020, doi: 10.32604/jqc.2020.010815.