HYBRID TIP SELECTION ALGORITHM IN IOTA

MAHSHID AGHANIA
Master of electrical engineering, KIAU, 2017

A thesis submitted
in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

HYBRID TIP SELECTION ALGORITHM IN IOTA

MAHSHID AGHANIA

Date of Defence: January 7, 2022

Dr. R. Benkoczi Thesis Supervisor	Associate Professor	Ph.D.
Dr. D. Gaur Thesis Examination Committee Member	Professor	Ph.D.
Dr. S. Hossein Thesis Examination Committee Member	Associate Professor	Ph.D.
Dr. W. Osborn Chair, Thesis Examination Committee	Associate Professor	Ph.D.

Dedication

This thesis is dedicated to my family. To my beloved parents Ali Aghania and Molok Maleki, and my brothers, Mohammad Aghania and Vahid Aghania who always believed in me and supported me with their love and passion.

Abstract

Distributed Ledger Technology (DLT) refers to the technical architecture that enables simultaneous access, validation, and record of transactions in an immutable way over a network. IOTA is a distributed ledger developed to record and send transactions between nodes in the Internet of Things (IoT) design. Node is an electronic device that can create, receive, or transmit transactions over the IOTA network, known as the tangle. Every node in IOTA wants to submit a transaction. The network will allow nodes to submit their transaction, only after they run the Tip Selection Algorithm (TSA). TSA is an essential part of the IOTA tangle. The term "tips" refers to transactions that are still waiting to be approved by other nodes. The unverified transactions are called orphan tips, meaning that orphan transactions are not approved by any node. Nodes in the tangle that approve older transactions are called lazy nodes, and transactions submitted by lazy nodes are called lazy tips. There should be a trade-off between verifying a transaction (orphan) and how quickly a transaction is verified (lazy tips). The importance of the TSA is to balance the number of orphan transactions and lazy tips. Our contribution in this thesis is to make adjustments in the TSAs by assigning adaptive values for the TSAs. Parameter α is a determining factor in the TSA algorithms to adjust the number of lazy and orphan tips. In this thesis, we propose a new hybrid TSA algorithm. The hybrid TSA employs a recursive walk with a variable α parameter. In the experimental analysis of the thesis, we measured the orphan and lazy tips for different TSA algorithms from the output data generated by the IOTA simulator. The result shows that the hybrid TSA can effectively eliminate the number of lazy tips.

Acknowledgments

I wish to express my sincere appreciation to my supervisor, professor Robert Benkoczi. He generously supported me from the first day I came to Lethbridge. He convincingly guided and encouraged me to be professional and do the right thing even when the road got tough. Without his persistent help, the goal of this project would not have been realized.

I want to thank Dr. Daya Gaur and Dr. Shahadat Hussain, my dissertation committee. They've been a great mentor. Without their continuous positive post, I would not have been ready to complete the dissertation.

I want to extend my sincere thanks to Dr. Kharaghani and Dr. Akbary, who have, mostly out of kindness, helped along the journey of my master.

I want to especially thank Dr. Muhammad Ali Khan and Jefferson Gardner. During the first 2 years of staying in Lethbridge, they persistently helped me to gain more enhanced knowledge in the practical field of blockchain and IOTA, They provided me many chances to share and exchange information, where it definitely encouraged me to expand my knowledge and creative thinking.

In the coding part of my thesis, I wish to express my deepest gratitude to Peter Nguyen, who is a genius programmer and Orville Lim, the most excellent Javascript mentor, who elevated my programming skills in javascript, especially on analyzing Javascript data that was a great asset for my thesis. Finally, I want to thank Ehsan Hamzei for troubleshooting and reviewing my code.

A very special thanks to Alon Gal, who helped me gain a more in-depth understanding of IOTA and the associated mathematical concepts.

I want to thank my colleague, Leila Karimi, for her great support and friendship during

my stay in Lethbridge, Akalanka Galappaththi, Anna Nikolova, Ajay Raj, Sharmin Akter and Nawrin Ferdowsi. They were all true friends of mine. A very special Thanks to Sajad Fathi and his precious wife, Zahra, for their greatest care.

I wish to acknowledge my brothers, Mohammad Aghania and Vahid Aghania. They kept me going on, and this work would not have been possible without their input.

The last word goes to my parents, Ali Aghania and Molok Maleki, who were the light of my life and gave me extra power and motivation to do things. This thesis is dedicated to them.

Contents

List of Tables

List of Figures

List of Symbols

Constants

α An adjustable parameter in the TSA indicating whether the TSA is weighted or unweighted

$Ł_0$ Total number of tip transaction

λ The arrival rate is the number of transaction's arrivals per unit of time.

σ Standard deviation of weights (STD)

$\mathbb{T}$ The current time

$\mathbb{T}_{Threshold}$ The maximum time that a transaction can wait to receive approval.

h Hidden time

SD Squared deviation of weights

V Transactions in the tangle (vertices in the graph)

W Transaction's weight

X New transactions arrival

Glossary of Terms

- **Node**: A computer that can send or receive transactions.
- **Transaction**: A message including text or value that is transferred among the nodes.
- V_{First}: First transaction/site submitted in the tangle
- V_{Last}: Last transaction/site submitted in the tangle
- **Ledger**: A ledger is a record of transactions stored by the nodes.
- **Distributed Ledger Technology (DLT)**: DLT is a distributed network where every node stores a local copy of the ledger without the need of the central authority.
- **Tangle**: A new type of DLT based on the Directed Acyclic Graph (DAG) data structure.
- **Tip**: A transaction in tangle that is waiting for its approval is called a tip transaction.
- **Sites**: Sites are representative of both nodes and transactions (in the simulation). Upon submitting a transaction by a node, a site is generated in the simulator (acting as a transaction). Every site in the simulator needs to approve two previous transactions (acting as the node).
- **Tip Selection Algorithm (TSA)**: Algorithms that nodes in the tangle use to select the tip transactions.
- **Cumulative weight**: The cumulative weight of a transaction is equal to the weight of the transaction itself plus the total weight of transactions approving the transaction.

- **Scalability**: Scalability is the capability of a network to validate more transactions (in IOTA) or blocks (in the blockchain). By default, the blockchain network is not scalable since transactions need to wait for the approval of the miner.

- **Proof of Work (PoW)**: Proof of work is solving a cryptographic puzzle by the miners in a distributed system so that miners can ensure the validity of the calculated hash.

- **Hidden time (latency)**: Latency or hidden time in the blockchain applications is related to the miners. Miners might postpone validating a block because of the latency in PoW. Hidden time in the IOTA network is caused by the participating nodes in the tangle doing the proof of work, or there might be some node delaying the whole tangle due to lower computation power or using slower TSAs.

- **Weight Coefficient Factor (WCF)**: A measurement factor that calculates the chance of a transaction being chosen by the TSA.

- **Coordinator**: The central authority in the current version of IOTA.

- **Milestone**: Transaction submitted by the coordinator. If a milestone transaction approves a transaction, all the previous transactions approved by the parent transaction will receive approvals.

- **Coordicide**: The IOTA version 2.00 that is completely decentralized and not using the coordinator as the central authority.

- **FPC**: The consensus algorithm in IOTA version 2.00.

- **Mana**: The security id in a transaction shows the pledges that a node has received.

Chapter 1

Introduction

Distributed Ledger Technology or DLT refers to the technical architecture that multiple users can access, share or have the record of transactions in an immutable way [41]. Nodes in IOTA are computer devices that can send and receive transactions from nodes in the tangle. Nodes are responsible for the following: approving a transaction and keeping transactions recorded in their local ledger. Nodes also broadcast the record to the rest of the nodes in the tangle. In order to approve a transaction, nodes need to find the encrypted data for each transaction. Every transaction information such as their timestamps, transaction history, transaction amount, senders and receivers information is encrypted using a hashing algorithm. Since the history of a transaction is included in the encrypted value, upon decrypting a transaction, the node will be aware of the history of the transaction and therefore altering a transaction is difficult. All nodes keep a record of the balances of addresses, so they can check that a transaction is not transferring more IOTA tokens than are available on the address. when a transaction is confirmed, can nodes update their record of balances. When any node in the tangle submits a transaction, they will first approve two previous transactions. Approving the previous transactions helps the tangle to increase transaction validity. Therefore transactions will get approvals faster, and nodes will no longer be required to wait for miners to accept the validity of their transactions.

In a distributed ledger such as IOTA, there might be some nodes trying to 'tamper' the ledger [37]. Tampering a ledger includes changing a transaction history, amount, date, balance. Ledgers in distributed applications are immutable, meaning that once a transaction

is added to the ledger, no malicious node can change the transaction data in the ledger [44]. In order to eliminate the malicious node to tamper the ledger, nodes will compete in solving a cryptographic puzzle to find the transaction data.

Every node in IOTA will run the Tip Selection Algorithm or TSA to validate transactions legitimacy. TSA refers to the methods that the nodes in IOTA employ to select unapproved transactions (tips) [22]. Approving a transaction indicates the transaction legitimacy so that the money transferring in a distributed network is reliable. If a transaction is unapproved (orphan), the content of transactions is not approved, and the payment is suspended.

The first goal of this thesis is to improve the TSA algorithm so that the number of orphan tips in the system is reduced. Elimination of orphan tips results in fewer on-hold transactions in the tangle and more users being encouraged to use the IOTA network with a more efficient TSA algorithm that helps their transactions be validated.

Second, some nodes in the tangle, known as the lazy node, are approving older transactions. Although the validation of an orphan transaction is important (so that transactions are no longer on hold), the transactions history must be trimmed depending on the storage of the tangle. As a result, the lazy nodes are causing issues for the tangle, and we want to get rid of the lazy nodes.

1.1 Problem definition

The main problem of the current TSA algorithms is the emergence of lazy tips as the result of the TSA algorithms as well as leaving orphan transactions in the tangle. The existence of lazy tips and orphan transactions can reduce the efficiency of the network by slowing down the transaction's approval speed so that users are less courageous to use IOTA for their micro-transactions. To address how the orphan transactions and lazy tips are generated in the tangle, we have the following definition in the tangle:

Definition 1.1 (**Snapshot**). A snapshot is a process of reducing the size of the tangle database, by removing transactions and restarting the tangle from a new transaction.

Definition 1.2 (**Orphan tips**). The orphan tip is transaction that has not received approvals upon the snapshot, meaning they have no parent transaction within the trimmed sub-tangle.

The problem with having multiple orphan tips in a tangle is that the tangle's transactions will remain on hold, preventing funds from being transferred in IOTA. On the other side, transactions must be trimmed because of the ledger's storage limit, therefore having a slow node changes the ledger's linear timestamp (an older transaction comes back to live), making it difficult for the coordinator to trim the tangle into smaller parts. The other problem in IOTA is the occurrence of lazy tips as the result of TSA. We define lazy tips as:

Definition 1.3 (**Lazy transaction**). A lazy transaction is submitted by a node that approves a transaction that has been waiting for approval for more than the specified amount of time. In our thesis, we refer to the IOTA foundation's mentioned specified time (we call it thresholds) as follows:

- In a network with a small transaction rate (λ=0.5) which is similar to IOTA 1.00 (legacy network), the waiting time threshold is 2-3 minutes [1].
- In a network with $\lambda = 15$ which is similar to IOTA 1.50 (Chrystal network), the waiting time threshold is around 9-10 seconds [1].
- In a network with with $\lambda = 20$ or more which is similar to IOTA 2.00 (Cordicide network [2]), the waiting time threshold is around 3 seconds [4].

Suppose transactions in the tangle are generated using a random event generator called the Poisson process model. A Poisson Process is a model for a series of events where the average time between events is known, but the exact timing of events is random.

In our thesis we use the Poisson point process to model transaction's arrival in tangle where λ is the expected rate of transactions occurrences (transactions per seconds) and t is the average inter-arrival time.

Assume X is the time when new transactions are generated/arrive in the tangle. If the time threshold is large enough, new transactions have a high probability (more than 99 percent) of being generated in the tangle within the time threshold ($P(X < T_{Threshold})$). If a transaction is "lucky", it receives approval by the same nodes submitting their transactions within the time threshold. We use the Poisson distribution to calculate the probability of new transactions occurring within the $T_{Threshold}$:

$$P(X < T_{Threshold}) = 1 - P(X > T_{Threshold}) \tag{1.1}$$

The Poisson probability that the new transaction arrival (X) is within the threshold range ($P(X < T_{Threshold})$), in a network with $\lambda = 0.5$ and $T_{Threshold} = 120$ second is:

$$P(X < T_{Threshold}) = 1 - e^{-\lambda t} = 1 - e^{-60} = 1 > 0.99 \tag{1.2}$$

If we model the IOTA network with $\lambda = 15$, the Poisson probability that new transactions arrive within ($T_{Threshold} = 9$ seconds) is:

$$P(X < T_{Threshold}) = 1 - e^{-\lambda t} = 1 - e^{-135} = 1 > 0.99 \tag{1.3}$$

Also, if we model the IOTA network with $\lambda = 20$, the Poisson probability that the transaction's arrival time (X) is no longer than the threshold ($T_{Threshold} = 3$ seconds) is:

$$P(X < T_{Threshold}) = 1 - e^{-\lambda t} = 1 - e^{-60} = 1 > 0.99 \tag{1.4}$$

We call a tip transaction an unlucky tip if there is more than a 99 percent chance that a new transaction arrives in the tangle within $T_{Threshold}$, but still, the tip transaction receives no approval. If the transaction is not lucky to receive an approval within $T_{Threshold}$, it may receive approval from a node that submits its transaction after $T_{Threshold}$. We refer to the approving node as a lazy node, and the transaction submitted by the lazy node is a lazy

transaction.

1.2 Contribution

The thesis contribution is as follow:

- Proposing a new "hybrid" TSA algorithm based on varying the random walk parameter (α) so that orphan and lazy transactions are less likely to happen in the system. In the hybrid TSA-1, the random value of α is changing every time the nodes run the TSA, therefore every tip transaction will get a new chance from each of the nodes running TSA. The hybrid random walker also changes the value of α each time the random walk traverse to a new transaction, therefore all the parent transaction of the following transaction will be assigned with a new random value of α. In hybrid TSA-2, all the tip transactions will receive the same random value of α upon running TSA by the nodes, however, while the random walker traverse recursively to the parent transaction of the current transaction, all the parent transactions will receive a new random value of α. The following change of α helps to reduce the number of orphan and lazy transactions.

- Simulate and compare the result of the STD TSA, measuring the number of orphans and lazy tips.

- Presenting the new definition of the threshold range for measuring lazy tips. The study of lazy tips (precisely the measurement of lazy tips) has not been done before in previous studies. Therefore, it's worth analyzing how the lazy nodes can affect the tangle's performance and the possible relation between orphan and lazy tips.

- Finding the orphan tips from the output ledger of the simulated tangle and we calculated what percentage of the total transactions are orphan. We want to conclude how the TSA algorithms, especially hybrid TSAs, can reduce or increase the orphan tips compared to the other TSA methods.

This thesis will study issues caused by lazy and orphan tips. Different tip selection methods are tested to determine how they impact lazy and orphan tips under different environments. Finally, the hybrid tip selection algorithm is assessed as an outcome of the proposed TSA.

1.3 Organization of Thesis

The thesis organization is as follows: Chapter 2 presents an overview of blockchain networks, followed by a review of related works on traditional blockchain and DAG-based IOTA data structure. The TSA algorithms are described in Chapter 3 of the thesis. Chapter 3 also discusses our proposed TSA algorithm and why the hybrid TSAs can reduce orphan and lazy tips. In continuation, Chapter 4 introduces the simulation and our experimental results. Finally, Chapter 5 summarises the thesis and suggests some possible future study paths.

Chapter 2

Background and Preliminaries

In this chapter, we will provide background information that will be useful in understanding the thesis.

2.1 Distributed ledger

Because of technological advances, a growing number of users choose to transfer and receive money online rather than in person at a bank. Online banking and money transfer systems are continuously updating database records managed by central authorities such as banks or government-run sectors [27].

Online banking requires a trusted third party to communicate between two unknown parties. If a problem happens, the third party solves the issue between the two parties. There are multiple problems along with using a third party to finalize the transactions. First of all, the third parties charge a fee for the services they provide. Second, it takes time for the transactions to be approved. For example, processing a credit card transaction can take up to 3 to 7 business days. The third issue is the concern about having complete control over the transactions. Central authority can invalidate any transaction at any point. Fourth, the third party can keep track of all of the transactions among different parties. Finally, because there is only one central authority, the entire system will collapse if the controlling center is hacked or compromised [28].

The mentioned problems can be solved by using Distributed Ledger Technology (DLT). DLT eliminates central authority and allows for direct peer-to-peer interactions. In DLTs,

there is little cost for the transaction's verification [27]. Transactions between two parts of the world happen in a matter of minutes. All the verified transactions are immutable, which means once a transaction is recorded in the ledger, it cannot be changed. Also, in DLTs, several copies of ledgers are kept by the nodes to make modifications. The attacker must compete to gain the trust of 66 percent of the nodes[24], which is challenging since unauthorized changes in the ledger would quickly notify other users.

One of the foremost DLT is Blockchain. Satoshi Nakamoto [29] invented the Bitcoin blockchain to establish the digital token known as Bitcoin. Bitcoin is the most popular type of blockchain technology. Bitcoin is a peer-to-peer network. Nodes in a bitcoin network can send and receive transactions by publishing digitally signed messages to the network. All the block's information such as sender's address, receiver's address, date, amount of payment, time stamping is hashed using the SHA-256 cryptographic hash algorithm [11].

Following are the blockchain components:

- **Node**: A user in blockchain architecture, and every user has their independent copy of the ledger.
- **Transaction**: multiple data records which include an amount of money, value, time stamp, date, etc.
- **Block**: A group of transactions. The number of transactions within a block depends on block storage.
- **Chain**: A group of blocks in sequential order.
- **Miners**: Nodes who find the block's hash.
- **PoW**: Finding the encrypted hash of a block is called Proof of Work (PoW) and miners will receive rewards if they complete PoW of a block successfully before the other miners solve the PoW.

SHA-256 output is fixed for any size of input text. Nodes will start from the first block with a nonce equal to zero to find the block's cryptographic hash. For the nodes to keep track of the blocks they hashed, every block in the blockchain will be defined with a specific number of nonce in the header [29]. The Nonce (number used once) is a unique identification number for each block. If the node finds the block hashes that has more than four zero digit, the node wins the competition. However, if the block hashes do not provide the circumstances (doesn't start with four zeros), then the node needs to increase the nonce number until it finds a block with a hash beginning with four zeros [16]. The process of finding a block with a hash that starts with a certain number of zeros is called mining, and the node that wins the competition is called miner.

Nodes who submit transactions in the blockchain will pay the transaction fee for their transactions to be verified, and miners will benefit from the transaction fees [29] [20]. However, because of greater transaction fees in blockchain, In the case of micropayments, a greater transaction fee than the payment makes no sense.

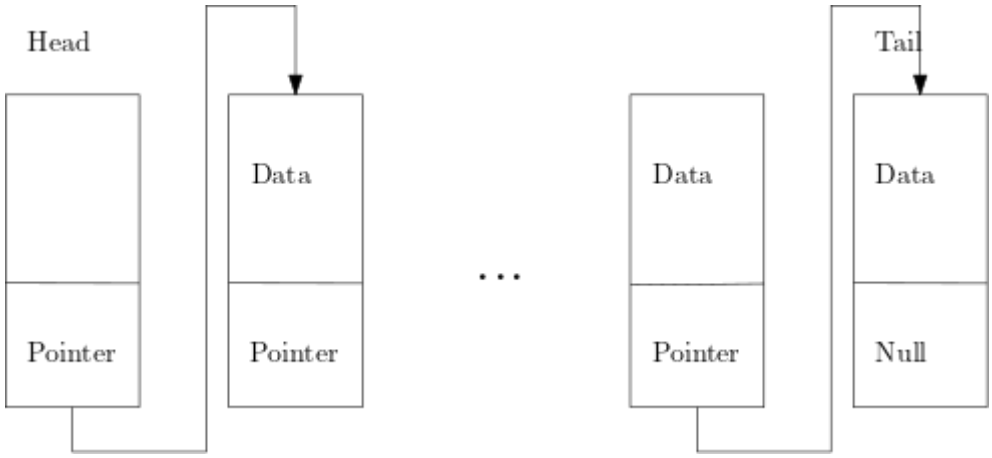


Figure 2.1: Blockchain structure

In the blockchain, a new block of multiple transactions attaches to the chain. Each block is linked to the previous block with a hash generated by the previous, and current block [31]. Every user on blockchain will update their ledger when a miner broadcasts the network's latest block. All users will not have the same chain picture in their local ledger,

as some users do not have a recently updated ledger [12]. Therefore, the node with the longest chain is the reference node and is considered the trusted node in the blockchain.

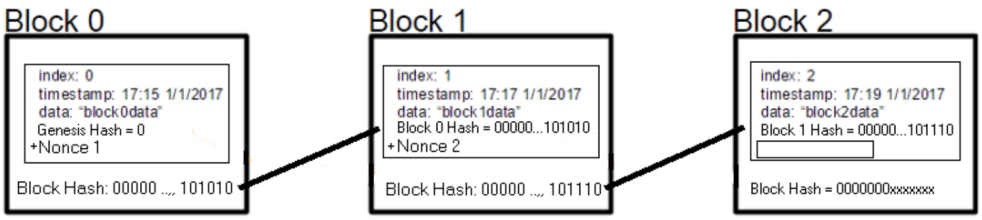


Figure 2.2: Blocks data

Miners are responsible for the latency in approving new blocks and, as a result, the blockchain's scalability issue [10][42]. Miners are given blocks of transactions and compete to solve a complex puzzle (finding the nonce number). Finally, the miner who solves the puzzle can add a new block to the chain and broadcast it to all the other nodes to update their ledgers [13].

When the miner solves a cryptographic (mathematical) puzzle, the transaction is considered validated. In particular, miners execute a Secure Hash Algorithm 256 bit (SHA-256) to generate an output known as a "hash." Blockchain rewards miners if they mine the blocks by finding the corresponding hash for each block [43].

Another parameter in the blockchain is block size and the number of transactions each block can store. Blocks with increased capacity help in the scalability of the blockchain because as more transactions are included in a block, more transactions are validated in single mining operation. After revealing the 6th block [17] in the chain, all transactions are considered validated. A mined block's status will remain unknown until the sixth block gets approval by the miner.

Blockchain architecture is created as follows:

1. A transaction is requested.

2. A block containing the transaction is created.

3. The block is sent to every node in the network, and miners start mining.

4. Miners find the block nonce.

5. Miners receive a reward for the mining

6. A block is added to the existing blockchain.

7. All the transactions within the block are approved.

8. The transactions are hidden in the system until six blocks pass onto the chain [17].

According to the white paper [29], when a large number of blocks are added to the Bitcoin network at once, the Bitcoin blockchain experiences scalability issues which means users need to wait a long time until their transaction is approved. In addition to the scalability issue, the transaction fee for each transaction submitted to the network is another Bitcoin blockchain issue. As a result, high transaction fees in micropayment transactions in IoT applications discourage users from using Bitcoin blockchain because a transaction fee greater than the value transferred is unreasonable. Until now, blockchain is unable to suggest lower transaction fees since the fees in the blockchain are the incentives for the miners.

Ledgers with a blockchain data structure also lack the storage that each node requires. For example, the blockchain ledger needs a capacity of 250 GB for a node to participate in the network, leading users to have Bitcoin blockchain size problems. Other blockchain networks also suffer from the same problem. The blockchain size of Ethereum has already crossed 1 TB. However, not every node is required to download the entire blockchain to participate in the network. Some nodes are known as full nodes. Full nodes are required for miners or users who prefer to participate in current transaction validation. Miners must therefore download the entire 250 GB blockchain data set. However, some users are just participating in a blockchain network to make transactions rather than to confirm them.

The Bitcoin blockchain is also vulnerable to quantum computers [39]. If the adversary node uses a quantum computer, it can mine new blocks in under 10 minutes and win the block sequence. In this case, the quantum computer is 1.25 times faster than the honest nodes doing the proof of work [19], and the malicious nodes can easily tamper the ledger.

Unlike blockchain ledgers, such as the Bitcoin blockchain, tangle uses a Directed Acyclic Graph (DAG) as its data structure for storing transactions. Tangle has the potential to solve some of the existing issues with traditional blockchains. First, the tangle network has no transaction fees, making it suitable for IoT micropayments [34]. Second, tangle is quantum-resistant [18]. Third, there are no restrictions on transaction arrival rates in the tangle network. Furthermore, there is no concept of mining in the sense that miners receive monetary rewards. There is no difference between transaction senders and transaction verifiers in the tangle network, as there is in traditional blockchain networks.

2.2 Directed Acyclic Graph (DAG)

Graphs are powerful data structures for modeling transactions or block in a distributed ledger [6], [45] and can help us compare models to determine the similarity of two models or find differences between them. Figure 2.3 shows a DAG ($G = (V, E)$), where $V = \{V_1, ..., V_N\}$ is a set of nodes and $E = \{E_1, ..., E_N\}$ is an ordered set of edges.

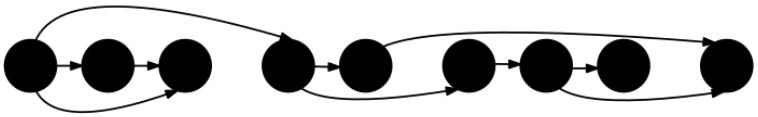


Figure 2.3: Directed Acyclic Grah (DAG)

DAGs data structure include:

- Directed edges (arrows)
- Vertices (transactions submitted by nodes)

- A path that is a sequence of transactions.

Suppose we have a direction of arrows from A to C. A, B and C are sequential transactions where B happens after A, and C happens after B. If a directed path from A to C exists, then A is an ancestor of C and C is a descendent of A.

$$A \rightarrow B \rightarrow C \tag{2.1}$$

Using this definition of A→B→C, A is a direct cause or parent of B, and B is a child of A and parent of C. At the same time, A is considered an indirect cause or ancestor of C. Node B is on the pathway from A and C.

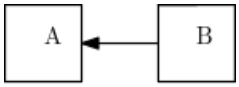


Figure 2.4: B directly approves A

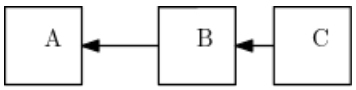


Figure 2.5: C indirectly approves A

DAGs are acyclic because there is no node pointing to itself, and all edges must contain arrows which means no directed path from any node to itself is allowed. If two nodes (and their submitting transactions) are not linked together directly or indirectly, the missing link shows that the nodes are not causally related.

2.3 Tangle components

The tangle main components are as follow:

Definition 2.1 (**Transaction**)**.** Transaction is a multiple data records that includes an amount of money, value, timestamp, date, etc.

Transactions are the basis of the tangle network. A transaction is a transfer from one account address to another address that can have a value or zero-value (contain data, a message, or a signature) [34]. Users as a node in the IOTA network will send transactions to tangle. If the transactions are verified, they are broadcast to all the other nodes. Nodes will update their ledgers to have the same transaction in their local ledger so that all the tangle nodes can have the same transaction. A transaction includes 2,673 encoded characters.

Nodes transmit transactions as a single entity known as a bundle. A bundle may include numerous transactions. When nodes in the tangle approve a bundle transaction, they approved all transactions in the bundle. Even if nodes want to broadcast a single transaction, they must prepare a bundle. Figure 2.6 shows a bundle transaction in IOTA that includes a head, a body, and a tail. The tail transaction is the one with a 0 in the current index field. As an example, transaction $TX(0,3)$ is the tail transaction in the index field of 3. The head transaction ($TX(3,3)$) is the one with the biggest value in the last index field and the last transaction in the bundle. The middle and tail transactions are connected through the field of the trunk transaction.

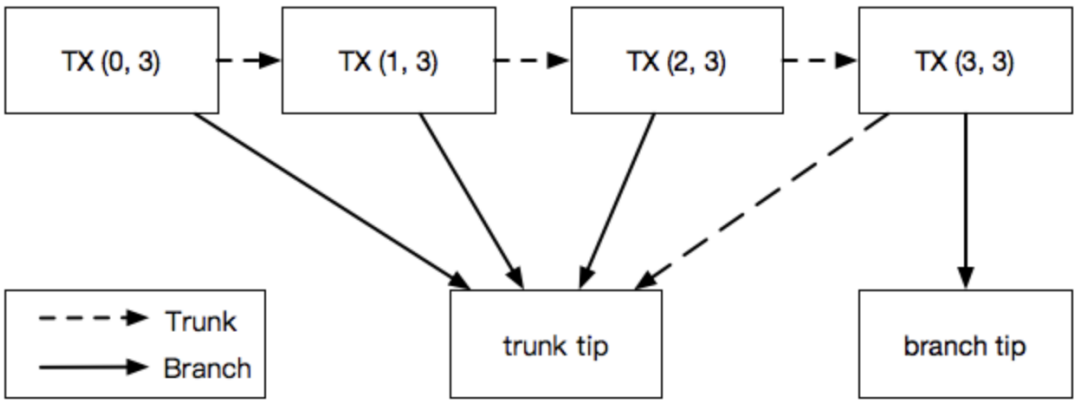


Figure 2.6: Bundle,Branch and trunk transactions

The following fields within a transaction bundle are:

1. Signature

2. Address

3. Timestamp transactions in order (First: Balance, Second: Sender, Third:Receiver)

4. Current index (Header index)

5. Last index (Tail index)

6. Value (0 for message, -1 for header,1 for tail)

7. Branch (When nodes run the TSA, the first link between two transaction is the branch)

8. Trunk (When the transaction selects the other transaction to verify, the second link between two transaction is the branch)

9. Hash of the bundle transaction

All transaction information such as an address, amount, time stamping, indexes, and the final index is stored in the bundle. Data in mentioned fields are immutable, which means the transaction records can never change. Suppose any change in the transaction's fields happens. In that case, the transaction hash will be different with any change in head, tail or middle transaction, leading to a wrong nonce number, making the transaction's children (all the other transactions that the invalid transaction issued) invalid. After approving previous transactions, nodes will append the hash value of the public and private keys and balances from the two prior transactions to its transaction, create a bundled transaction, and send it out to the network to be reviewed and validated by the next node that publishes a transaction. All the nodes in tangle need to have the same picture of tangle. Therefore, every participant node is required to have the most updated ledger.

Definition 2.2 (**Nodes**)**.** Node is a computer in the IOTA network. Node's role is to approve new transactions and approve transactions that are already existing.

Nodes are the heart of the tangle network. Every node in tangle has a copy of the ledger, including DAG information (transactions and their approval time). Nodes can read

information from DAG (view tip transaction), and they can write information to the DAG (publishing a new transaction).

Definition 2.3 (**Hashing algorithm**)**.** Hashing algorithms are cryptographic methods that accept a set of data as an input and generate a one-way encrypted hash of that data as an output.

Definition 2.4 (**Proof of Work (PoW)**)**.** Proof of Work (PoW) is a piece of data generated by nodes that meet certain criteria. PoW uses the Hashcash algorithm [30] to discover the encrypted piece of data (transactions address, timestamps and balances). Hashcash repeatedly hashes the same data over, with slight variations in the nonce number (an arbitrary number that can be used just once in a cryptographic communication), until a hash is found with a certain number of leading zero digits (more than four zero digits).

Definition 2.5 (**Miners**)**.** Nodes who run the PoW are called the miners. In IOTA, every node also acts as a miner and verify transactions validity.

Definition 2.6 (**Tips**)**.** Tips are defined as transactions submitted by nodes.

Definition 2.7 (**TSA**)**.** Tip Selection Algorithm (TSA) is the algorithm to select tips. TSA is run by each node two times.

Definition 2.8 (**Hidden time**)**.** : Hidden time in IOTA is the latency that nodes experience. Every node in IOTA is waiting for the earlier nodes to submit their transactions. The previous nodes, however, might have lower computation power in running PoW causing a delay in finding the transaction's hash. Also, if nodes postpone running the TSA, which results in finding a tip transaction with delay, the validation of a transaction may take a longer time. All the mentioned reasons can cause delay known as the hidden time in the TSA.

Definition 2.9 (**Cumulative weight**)**.** Cumulative weight is defined as a transaction's own weight plus the sum of weights of all transactions that directly or indirectly approve this transaction [36].

Equation 2.2 shows the cumulative weight of a transaction, where $H(v)$ is the total weight of all the transactions that approve v. In other words, the cumulative weight of a vertex is the number of ancestors it has plus one.

$$F(v) = 1 + H(v) \tag{2.2}$$

Figure 2.7 shows the the cumulative weight of a finite DAG including 7 nodes publishing their transaction $V_1, ..., V_7$.

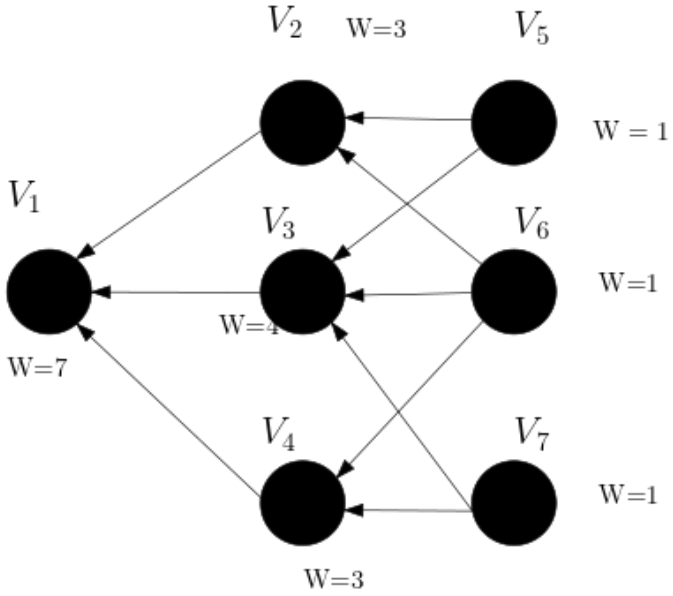


Figure 2.7: Cumulative weight

Definition 2.10 (Consensus). When nodes make the final decision on the validity of a transaction, they reach a consensus on whether to accept or reject a block

Nodes also are responsible for deciding which transactions to confirm as the unapproved transactions remain pending. Nodes will attach transactions to their local ledger upon approving a transaction. We can consider that nodes in the tangle have the same responsibility as the miners in the blockchain. In contrast to the blockchain, not every user can be a miner since blockchain mining blocks demand huge electricity and computing power.

In IOTA, there might be times where different nodes might have different opinions on

a transaction [35]. The current version of IOTA is not completely decentralized and uses a central node called Coordinator to verify transactions.

Definition 2.11 (**Coordinator**)**.** The Coordinator is a powerful node operated by the IOTA Foundation that issues different types of transactions (called **milestones**) with zero values. Milestone transactions only have a message (they do not have any value since they are not transferring money) to validate transactions. If a milestone transaction points to a transaction, all the other transactions approved directly or indirectly by the milestone transaction are considered approved (red vertices).

Figure 2.8 shows the coordinator node publishes the milestone transaction (red vertex). Upon approval of the tip transactions, the milestone transaction will link to the approved transactions.

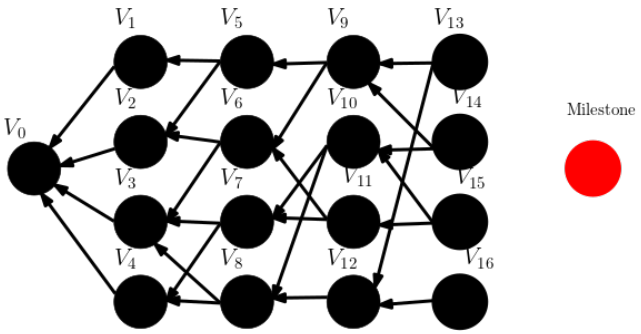


Figure 2.8: Milestone transaction is submitted by the coordinator

Figure 2.9 shows when a milestone transaction links to a tip transaction, the coordinator approves all prior transactions that were linked by the tip transaction.

The most recent version of IOTA (IOTA 2.00) is completely decentralized, meaning that there is no coordinator or central authority submitting milestones to verify transactions. IOTA 2.00 uses **mana** to add a security code to each node. Mana is a measure of a node's reputation. The more trustworthy a node is, the more reputation/mana it will receive. Nodes can earn mana by processing transactions. The IOTA foundation will then give nodes mana as their incentives [15].

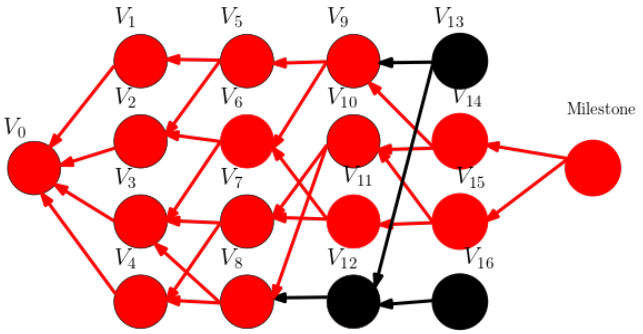


Figure 2.9: Every transaction that is linked by the milestone transaction directly or indirectly, is approved by the coordinator.

For the nodes to reach a consensus after mana, a voting algorithm in IOTA will ask the opinion of the majority of nodes randomly. The new consensus algorithm in IOTA 2.00 is called Fast Probabilistic Consensus (FPC) and is defines as:

Definition 2.12 (**FPC**)**.** FPC is a protocol that allows IOTA nodes to quickly reach an agreement on which of two conflicting transactions should be accepted by asking the opinion of neighbour nodes. If the neighbour random nodes agree with the validity of a transaction, they vote 1, which means they approve the transaction, otherwise, they vote 0.

IOTA 2.0 has been efficiently eliminating the possibility of the malicious nodes tampering ledger. Suppose a node receives two conflicting transactions. Using FPC, nodes can quickly find consensus and process transactions.

2.4 Tangle as a DAG data structure

The IOTA tangle uses Directed Acyclic Graph (DAG) as the data structure [26]. DAG is developing as an alternative to conventional blockchain architectures for distributed ledger technology [6].

Every node in tangle can publish transactions to the network and act as a user or issue the previous transactions and act as the miner. A node must do the following for any incoming transaction to be valid:

- Every node runs **TSA** for selecting tips in the tangle. The tip transactions selected by the TSA are the vertices of the DAG.

- Nodes ensure that there are no conflicts between transactions when they approve a transaction. The process of approving transactions, checking nonce numbers, and calculating transaction balances is similar to the **PoW** in blockchain.

2.5 Related works on distributed ledger simulators

Stoykov [40] studied a Bitcoin-like blockchain with the capability of large-scale network simulations, focusing on scalability and speed. The study also used a Java-based simulator for analyzing the scalability of Bitcoin-style blockchains. The research addresses various mechanisms that can be employed to resolve the scalability limitation of Bitcoin-style blockchains by measuring the differences between the simulator and real time scenarios.

Another study by Kreku [21] employed a simulator to optimize network latency in Ethereum. ABSOLUT was the simulation tool they used to measure the performance and power consumption. These simulations assume that the network latency between the nodes is minimal (less than 1 msec). They show that it is best to use the least number of nodes to satisfy the performance requirements.

The evaluation of selfish mining and the latency on blockchain application was developed by Gobel [17] using discrete-event blockchain simulators under a network with latency in sending decisions among the network miners. The simulation result of selfish mining indicates that selfish mining puts honest nodes in a position where they are wasting resources on mining blocks that have no chance of being included in the long-term blockchain.

Kreku [21] study shows the practical application of architectural performance modelling and simulation tools to predict blockchain latency. They also discussed how to leverage the simulator to improve decision-making in blockchain applications.

As a result of growing interest in DAG applications, several studies have tried to simulate the DAG ledger. Tools and visualizing models have been offering simulations for building DAG systems. Kusmierz [23] modelled a discrete simulator to discuss the relationship between cumulative weights of transactions and the number of tips that result in the unsatisfactory performance of the weighted random walk.

Lathif [25] suggested a DAG simulator with an adjustable parameters framework. The simulator is based on VIBES to model simulations with a large number of transactions. The drawback of the VIBE simulator is that the author still did not validate the simulator model.

Alon Gol [3] also developed a simulator to study the number of tips. The study shows effecting factor on the number of lazy tips is the cumulative weight, and as the cumulative weight increases, fewer lazy tips will happen in the system. However, the weighted TSA will result in more transactions left unvalidated. An IOTA simulator is developed for tangle visualization to observe several tangles generated with different tip selection algorithms such as uniform TSA, unweighted, random walk, and weighted random walk.

The adaptive random walker study [9] presented a hybrid TSA based on a simulator from Gol [3] that has the benefits of both unweighted random walk and weighted random walk at the same time. The authors claim that an adaptive random walker can approve tip transactions in a shorter amount of time. The lazy tips do not have the chance to be generated in the system, and a weighted algorithm can better resist lazy tips.

Ferrero's study used Gol's simulator to create an adaptive TSA so that when the node runs the TSA for the first time, Ferrero's algorithm selects a tip using a weighted TSA, and when the same node runs the TSA for the second time, Ferrero's algorithm employs an unweighted TSA. Ferrero's results indicate a significant decrease in the number of orphan tips.

The goal of our thesis is to improve existing TSAs by proposing a new TSA called hybrid TSA. We used the Gol simulator to measure the number of orphan transactions in the hybrid TSA, as well as the number of lazy tips, and we measured the performance of

different TSAs by presenting a ratio between lazy tips to orphan transactions in different TSA algorithms.

Chapter 3

Efficient Implementation of Tip Selection Algorithm

In this chapter, we explain different TSA algorithms using the simulator concept. The following sections help to get a better understanding of the simulator. Based on the simulation model, we describe how different TSAs can cause lazy and orphan transactions to occur in the tangle.

3.1 Introduction to the simulation

IOTA visualization [3] is a tool for modelling tangle's behaviour using the JavaScript statistics library [7] and React [38]. The following section describes how the simulation works by examining how a transaction is created using the Poisson process. As described in Chapter 2, a node must choose two previous transactions submitted by prior nodes.

When nodes submit a transaction in the tangle, the transaction is held until other nodes approve it. The unverified transactions will remain as a tip until the other nodes run the TSA algorithm and choose the unverified transaction. Every time nodes run the TSA to pick two transactions to verify.

Consider the total number of tips in the tangle, which is denoted as L_0 where $L_0 > 0$. The number of tips includes the number of tips revealed to nodes and the number of hidden tips [33]. The hidden time, abbreviated as h, is the amount of time that a node that publishes transactions must wait until their transaction is visible to the rest of the tangle. Also, at any given moment t, there are hidden tips that are not yet visible to the network. Assume that

there are typical r known tips (attached before time t-h and remain tips at time t). Let λ be the rate of the Poisson process. For any time t, there are specific hidden tips λh connected to the last r tips, but they are not yet visible to the nodes [5]. If r is the number of known tips at time t-h, the expected number of transactions in the tangle is [33]:

$$L_0 = r + \lambda * h \tag{3.1}$$

The DAG linear model is defined as a model to describe transaction arrival in a way that transactions arrive at every integer value of time. In order to simulate the transaction's arrival in the real tangle, we need to implement the non-linear simulator using the Poisson Process model. A Poisson Process is a model for a series of discrete events where the average time between events is known but the precise timing is uncertain. The arrival of an event is independent of the previous event. Poisson process model is suitable for simulating the arrival time of transactions. The JavaScript statistical library can return a random number with exponential distribution, and the arrival rate of λ [7], [3].

A Poisson Process has the following conditions:

- Every event is independent of another event.
- The average rate (events per second) remains constant.
- There are no two events that happen at the same moment.

To implement the DAG, the simulator implements the vertice and the links. Each vertex in a DAG is called a site. Each site keeps a transaction and its corresponding cumulative weight in the local version of the DAG. Every site will act as the node and upon arrival to the tangle, it needs all the TSA to select two previous sites. Therefore, sites are the representatives for both nodes and the transactions (tips). Sites in the simulation will call the TSA while at the same time they also need to be selected by the TSA and not left behind (being an orphan)

Each simulation run is as follows: first, sites are created using an exponential function. Incoming sites arrival times are sampled from an exponential distribution based on the value of λ for every incoming site. Every site in the tangle runs the TSA to approve two previous site. Let h be the average time for a site to complete the computations (PoW) needed to validate a transaction.

For the entire TSA run with 500 nodes, we assume that λ is constant during the simulation time. We assume that all devices have the same amount of computing power and latency (1 sec), and each transaction's weight is equal to 1.

3.2 TSA Algorithm Background

Assume that $t = 0$ is the time origin that the genesis transaction enters the tangle and $t = T$ is the current moment. When nodes at T run the TSA algorithm, the simulator separates all tip transactions in $(T - h)$, where h is the hidden time due to network latency.

- The hidden time (h) is the amount of time that recent nodes and their submitted transactions in the tangle are not visible to other nodes due to proof of work latency.
- As shown in table 3.1, $T - h$ is defined as the period that the TSA's random walk is allowed to find the tip transaction.

Table 3.1: Periods in the tangle

Period	Description
T	Revealed nodes and hidden nodes
h	Nodes that are running TSA and doing PoW are hidden to other nodes
T-h	The random walker finds the tip transaction in this period

Figure 3.1 shows node V_4 running TSA. Suppose t is the average inter-arrival rate of the transactions. In our example, $V4$ arrives t seconds after V_1, V_2 and V_3.

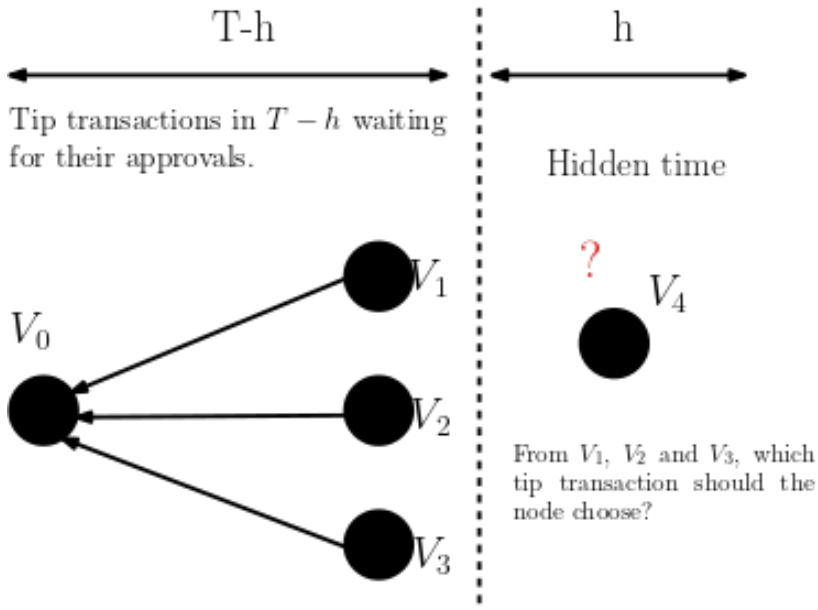


Figure 3.1: Node in the hidden time (node V_4 runs the TSA to choose a tip transaction in $T-h$)

If a tip is never verified and turns to an orphan tip, then the on-hold transactions will be declined. If the orphan tips increase, declined transactions will grow in the tangle. As more users' transactions decline, the users will lose the courage to submit their transactions with IOTA because more users will receive failure messages for their submitted transactions.

We can conclude that the trade-off between the orphan tips and the lazy tips to analyze the performance of a tip selection algorithm.

3.3 Random walks in tangle

Figure 3.2 shows the random walk steps. The random walker traverses the tangle from the first submitted transaction in $T-h$, until finding a tip transaction in $T-h$ that needs to be approved by the node in h [8].

Following are the random walk steps:

- Start with the first transaction.
- Find transaction's approvers.

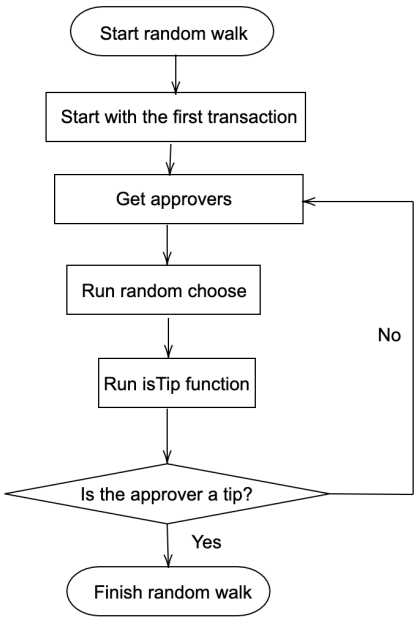


Figure 3.2: Random walk in tangle

- Choose a random transaction among the approvers and traverse to the chosen transaction.
- Check if the approver (chosen transaction in step 3) is a tip transaction.
- If the transaction is not a tip, the random walker will continue to find the approver's approvers (random walker uses approver function in step 2 to find the approvers of the chosen transaction in step 3).
- Randomly choose one of the approvers and check whether it is a tip.

By running the approver function on a transaction, the random walk will get all the parents of the selected transaction. Approver's function begins by obtaining all transactions and corresponding links until $T - h$. The following example explains the approver's function more clearly:

As it is shown in figure 3.3, let's say we have V_0 and V_1 as well as L_0, L_1, and L_2. The approver function will begin with V_0 and search for links that lead to transaction V_0, that are L_0, L_1, and L_2. The approver function will associate these links with their source values,

which are V_1 for L_0, V_2 for L_1, and V_3 for L_2. Therefore, V_0's approvers are V_1, V_2, and V_3.

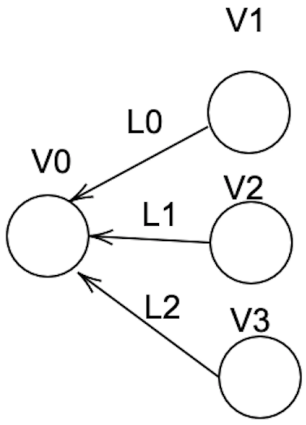


Figure 3.3: Approvers for transaction V_0 are (V_1, V_2 and V_3). L_0, L_1, L_2 target is V_0 and their corresponding sources is V_1, V_2 and V_3

Figure 3.4 shows the choose function in the random walk. In the choose function, the length of the approvers is determined first. Then a random number between the minimum and the maximum length of the approvers will determine the index of the chosen transaction among the approvers. In the previous example, V_1, V_2 and V_3 were the approver transactions. The upper and the lower bound is $0 \leq random \leq 2$, therefore the random chosen number can be 0 (index 0, transaction V_1), 1 (index 1, transaction V_2) or (index 2, transaction V_3).

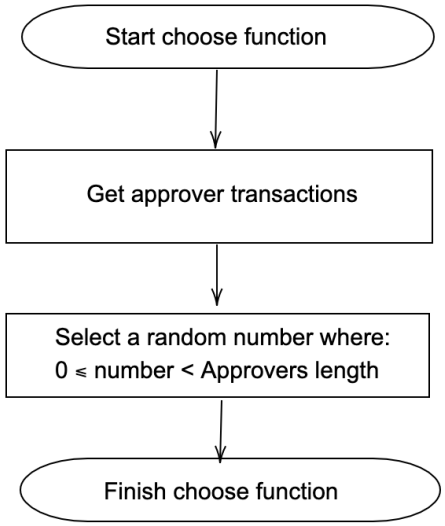


Figure 3.4: Choose function in random walk

The random walker's **isTip** function ensures that the approver transaction in $T-h$ is still tip (if the approver transaction has no parent transaction, the approver is a tip transaction). Suppose the current approver in $T-h$ is not a tip. In that case, the walker will continue to get the approver's approvers. The random walker will then choose a transaction among the approvers and walk/traverse toward the transaction. If the new transaction is a tip, the random walker will stop.

Figure 3.5 shows a random walk starting from V_0. The random walker begins with the starting transaction V_0 and finds approvers for transaction V_0. Transactions V_1, V_2, and V_3 are the approvers for transaction V_0. Random walkers will then randomly choose one of the approvers (choose function). Assume the choose function picks V_1. V_1 then must be examined to see if it is a tip. Since V_1 is a tip, the random walker stops.

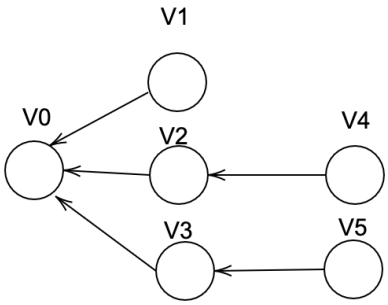


Figure 3.5: The random walker starts from V_0 and search for the tips (V_1, V_4, V_5)

3.4 Uniform TSA [32]

In the uniform TSA, the probability of selecting a tip is shown in equation (3.2):

$$P_{uniform} = \frac{1}{\sum_{t=0}^{T-h} Tips} \tag{3.2}$$

As tangle expands, the $T-h$ period contains more tip transactions. Because the chance

of selection is the same for all tips in $T - h$ [32], tips gain an opportunity to be selected by a lazy node instead of newly published tips.

3.5 Unweighted TSA

The unweighted random walk is similar to the uniform TSA. The only difference between the unweighted TSA and uniform TSA is that the unweighted TSA uses a random walker [32].

As mentioned in section 3.3, the random walker is a recursive function starting from the genesis transaction. It moves recursively to find the approver transactions and checks if they are still tips (not selected by other transactions).

Figure 3.6 shows an example of DAG with the unweighted TSA. Suppose the average inter-arrival time between transactions is one second. In our example in figure 3.6, V_1 is an tip transactions. Starting from V_0, the random walker select one transaction among the approvers (V_1, V_2, V_3, V_4, V_5). The probability of V_1 being selected by the TSA is equal to the probability of V_2, V_3, V_4 or V_5 being selected by the TSA. Therefore, the unweighted TSA can select the tip because any outgoing links visited by the walker have an equal chance of being picked. In our example, node V_{23}, picks V_1. Since (V_{23}) links to an older tip, V_{23} is a lazy transaction. To conclude, the lazy transaction appears in the tangle as the result of the unweighted TSA.

3.6 Weighted TSA

The weighted random walker starts from the first transaction, as shown in figure 3.7, and gets the approver transactions (approver function mentioned in the random walk section). The random walk will then select one transaction among the approvers based on the Weight Coefficient Factor (WCF). The WCF measurement [34], determines whether a transaction has a chance of being selected by the TSA and is calculated as follows:

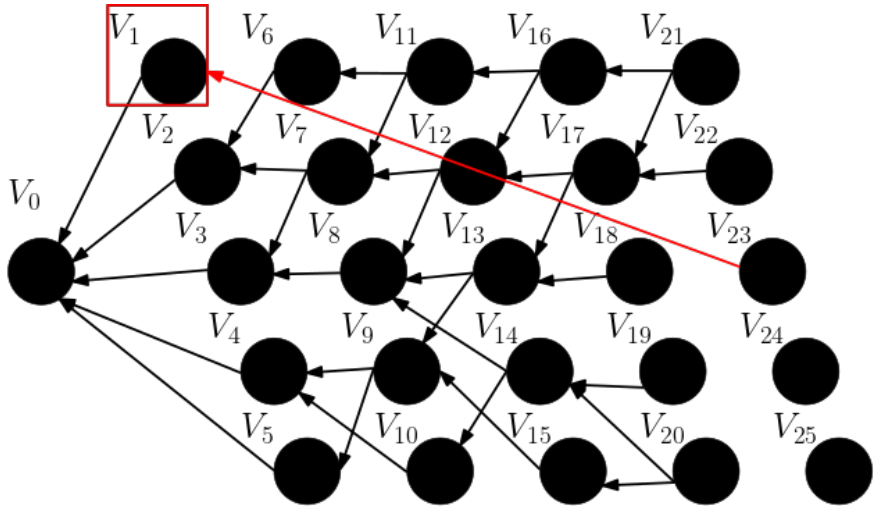


Figure 3.6: When the lazy transaction is appeared in the tangle as the result of unweighted TSA picking (V_1)

$$WCF = e^{-\alpha * W_{normalised}} \tag{3.3}$$

Parameter α in equation 3.3 is defined to change the TSA's weighting impact where the α upper and lower bound is:

$$[\alpha_{min}, \alpha_{max}] = [0, 1] \tag{3.4}$$

As a result, if the α value in equation 3.4 is equal to zero, the TSA behaves similarly to the unweighted TSA. On the other hand, if α is equal to one, the TSA behaviour is similar to the weighted random walk.

$W_{normalised}$ in equation 3.3 is the normalised cumulative weight. The cumulative weight of a transactions is the same equation we already explained in chapter 2 (2.2), where $H(v)$ that is the total weight of all the transactions that approve v.

To normalize the cumulative weights we have:

$$W_{Normalized} = W - W_{Max} \tag{3.5}$$

Figure 3.8 shows an example of seven transactions in the DAG named as V_0 (genesis

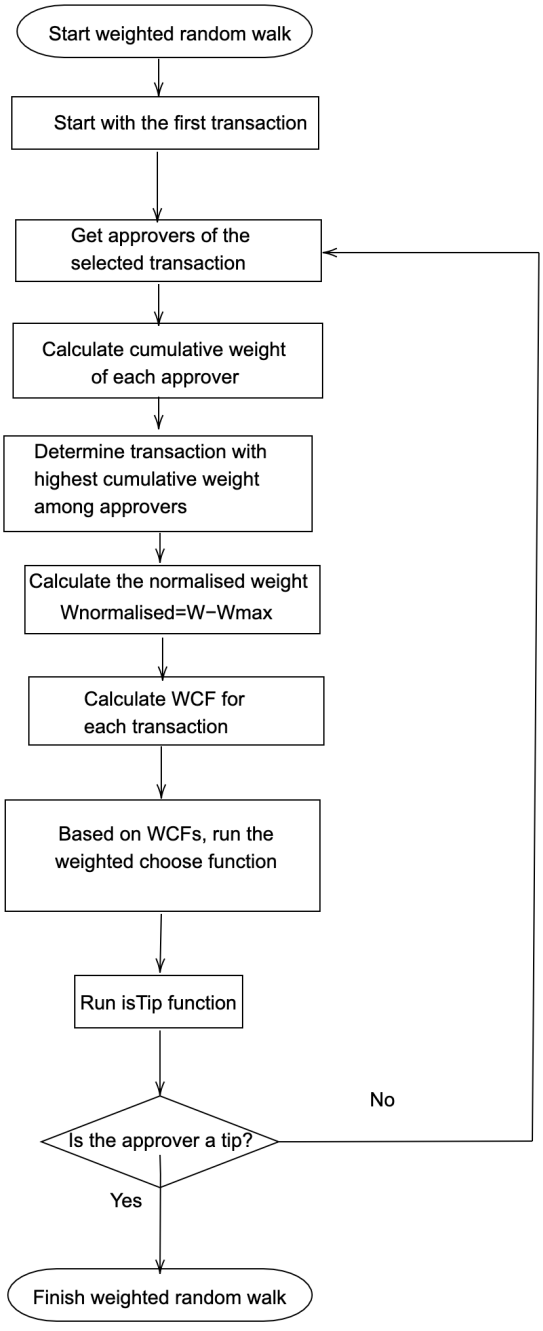


Figure 3.7: Weighted TSA algorithm

transaction),V_1,V_2,V_3,V_4,V_5,V_6,V_7. The corresponding node of $[V_5, V_6, V_7]$ will run the TSA algorithm six times. The weighted TSA is as follow:

1. The walker starts from V_0 and search approvers.
2. The approver of V_0 is V_1 which is not a tip, so the random walker continues looking at V_1's approvers that are $[V_2, V_3, V_4]$.
3. V_1's approvers ($[V_2, V_3, V_4]$) are still tips. The random walker stops walking.
4. The choose function decides to select among V_0's approvers ($[V_2, V_3, V_4]$). The WCF of each transaction is calculated.
5. Since α is the same for all the elements in ($[V_2, V_3, V_4]$) and they have the cumulative weight of 1, TSA is similar to the unweighted random walk ($W - W_{max} = 0$).

$$\begin{pmatrix} w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} w_2 - w_{max} \\ w_3 - w_{max} \\ w_4 - w_{max} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.6}$$

$$\begin{pmatrix} WCF_2 \\ WCF_3 \\ WCF_4 \end{pmatrix} = \begin{pmatrix} e^{-\alpha * w_2} \\ e^{-\alpha * w_3} \\ e^{-\alpha * w_4} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \tag{3.7}$$

Every element in $[V_2, V_3, V_4]$ has a WCF equal to 1. After calculating each transaction's WCF, the total sum of the WCFs as mentioned in figure 3.9 is calculated:

$$\sum WCF = WCF_2 + WCF_3 + WCF_4 = 3 \tag{3.8}$$

The probability of each transaction selected by the TSA is:

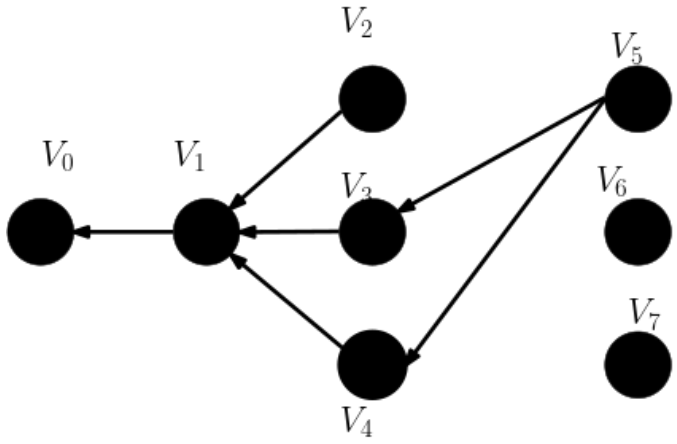


Figure 3.8: Weighted TSA(1) and TSA(2)

$$Probability = \begin{pmatrix} \frac{WCF_2}{\sum WCF} \\ \frac{WCF_3}{\sum WCF} \\ \frac{WCF_4}{\sum WCF} \end{pmatrix} = \begin{pmatrix} 0.33 \\ 0.33 \\ 0.33 \end{pmatrix} \tag{3.9}$$

The direct arrows from V_5 to V_4 and V_3 shows that V_4 and V_3 are chosen by the TSA:

$$TSA(1) : V_5 \rightarrow V_3 \rightarrow V_1 \rightarrow V_0 \tag{3.10}$$

$$TSA(2) : V_5 \rightarrow V_4 \rightarrow V_1 \rightarrow V_0 \tag{3.11}$$

The updated cumulative weight will be:

$$\begin{pmatrix} w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix} \Rightarrow \begin{pmatrix} w_2 - w_{max} \\ w_3 - w_{max} \\ w_4 - w_{max} \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} \tag{3.12}$$

So the WCF matrix for $[V_5, V_6, V_7]$ is updated as:

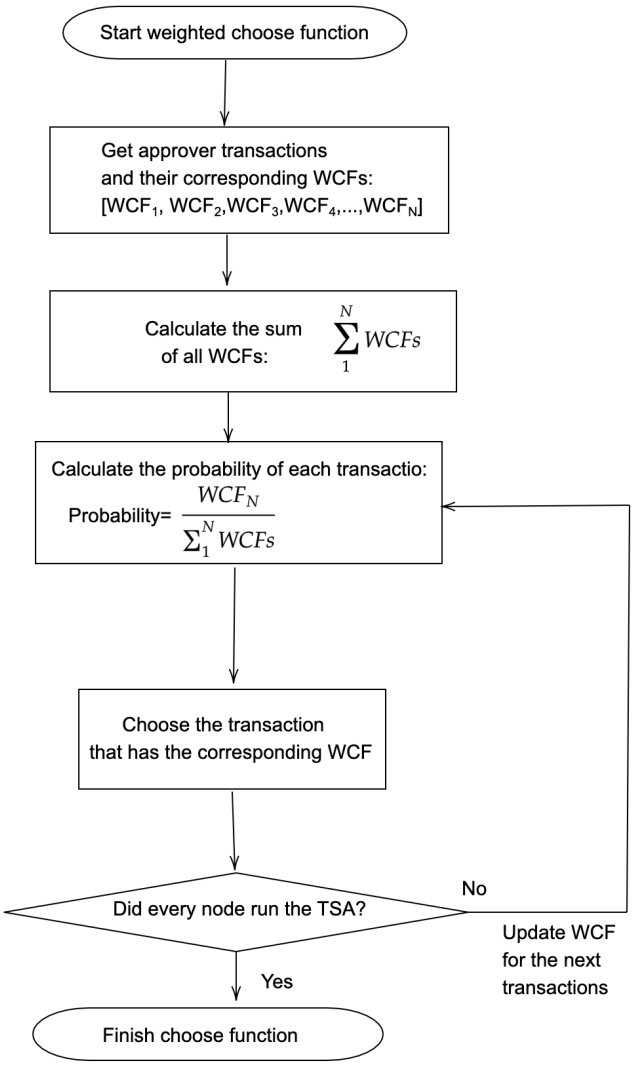


Figure 3.9: Weighted choose function

$$\begin{pmatrix} WCF_2 \\ WCF_3 \\ WCF_4 \end{pmatrix} = \begin{pmatrix} e^{-\alpha * w_2} \\ e^{-\alpha * w_3} \\ e^{-\alpha * w_4} \end{pmatrix} = \begin{pmatrix} 0.36 \\ 1 \\ 1 \end{pmatrix} \tag{3.13}$$

The total sum of the WCFs as mentioned in (figure 3.9) is calculated:

$$\sum WCF = WCF_2 + WCF_3 + WCF_4 = 2.36 \tag{3.14}$$

The probability of each transaction selected by the TSA is:

$$Probability = \begin{pmatrix} \frac{WCF_2}{\sum WCF} \\ \frac{WCF_3}{\sum WCF} \\ \frac{WCF_4}{\sum WCF} \end{pmatrix} = \begin{pmatrix} 0.15 \\ 0.42 \\ 0.42 \end{pmatrix} \tag{3.15}$$

Therefore, V_3 and V_4 are chosen by the TSA for the second time:

$$TSA(3) : V_6 \rightarrow V_3 \rightarrow V_1 \rightarrow V_0 \tag{3.16}$$

$$TSA(4) : V_6 \rightarrow V_4 \rightarrow V_1 \rightarrow V_0 \tag{3.17}$$

The updated coefficients are:

$$\begin{pmatrix} w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix} \Rightarrow \begin{pmatrix} w_2 - w_{max} \\ w_3 - w_{max} \\ w_4 - w_{max} \end{pmatrix} = \begin{pmatrix} -2 \\ 0 \\ 0 \end{pmatrix} \tag{3.18}$$

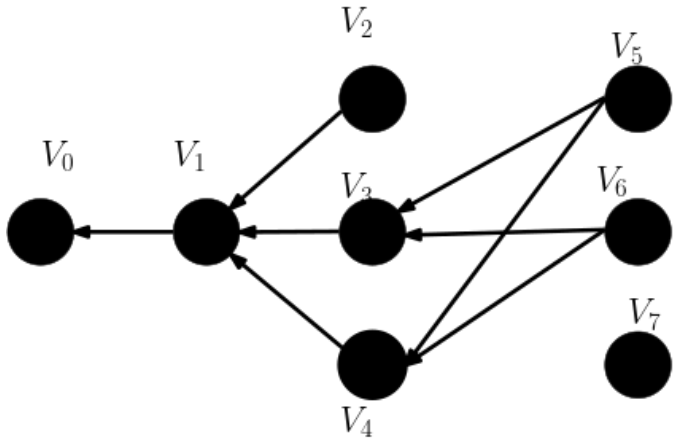


Figure 3.10: Weighted TSA (3) and TSA (4)

$$\begin{pmatrix} WCF_2 \\ WCF_3 \\ WCF_4 \end{pmatrix} = \begin{pmatrix} e^{-\alpha * w_2} \\ e^{-\alpha * w_3} \\ e^{-\alpha * w_4} \end{pmatrix} = \begin{pmatrix} 0.135 \\ 1 \\ 1 \end{pmatrix} \tag{3.19}$$

The total cumulative weight is calculated as:

$$\sum WCF = WCF_2 + WCF_3 + WCF_4 = 0.135 + 1 + 1 = 2.135 \tag{3.20}$$

The probability of each transaction selected by the TSA is:

$$Probability = \begin{pmatrix} \frac{WCF_2}{\sum WCF} \\ \frac{WCF_3}{\sum WCF} \\ \frac{WCF_4}{\sum WCF} \end{pmatrix} = \begin{pmatrix} 0.063 \\ 0.46 \\ 0.46 \end{pmatrix} \tag{3.21}$$

$$TSA(5) : V_7 \rightarrow V_3 \rightarrow V_1 \rightarrow V_0 \tag{3.22}$$

$$TSA(6) : V_7 \rightarrow V_4 \rightarrow V_1 \rightarrow V_0 \tag{3.23}$$

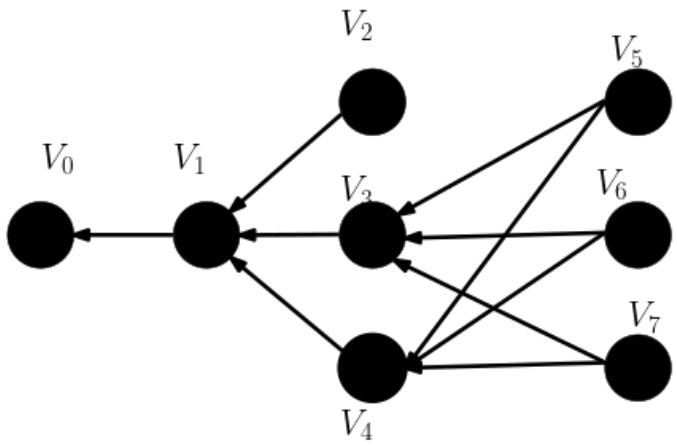


Figure 3.11: Weighted TSA(5) and TSA (6)

$$\begin{pmatrix} w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 4 \\ 4 \end{pmatrix} \Rightarrow \begin{pmatrix} w_2 - w_{max} \\ w_3 - w_{max} \\ w_4 - w_{max} \end{pmatrix} = \begin{pmatrix} -3 \\ 0 \\ 0 \end{pmatrix} \tag{3.24}$$

$$\begin{pmatrix} WCF_2 \\ WCF_3 \\ WCF_4 \end{pmatrix} = \begin{pmatrix} e^{-\alpha * w_2} \\ e^{-\alpha * w_3} \\ e^{-\alpha * w_4} \end{pmatrix} = \begin{pmatrix} 0.049 \\ 1 \\ 1 \end{pmatrix} \tag{3.25}$$

Therefore the WCFs are updated:

$$\sum WCF = WCF_1 + WCF_2 + WCF_3 = 0.049 + 1 + 1 = 2.049 \tag{3.26}$$

The probability of each transaction selected by the TSA is:

$$Probability = \begin{pmatrix} \frac{WCF_2}{\sum WCF} \\ \frac{WCF_3}{\sum WCF} \\ \frac{WCF_4}{\sum WCF} \end{pmatrix} = \begin{pmatrix} 0.023 \\ 0.48 \\ 0.48 \end{pmatrix} \tag{3.27}$$

V_2 will not get the approval and remain orphan. The mentioned example shows that tips can easily turn into lazy tips (V_2 in the previous example).

The weighted TSA can generate more orphan tips in comparison to the unweighted TSA. However, in the unweighted TSA, there is the chance that lazy nodes will approve the older tips, while in the weighted TSA,the older tips remain orphaned.

3.7 Standard Deviation Algorithm (STD) [9]

When nodes run the weighted TSA algorithm, the algorithm uses a fixed α value in the weighted random walks. Therefore, only the cumulative weight of the approvers is affecting the WCF. When a transaction has a larger cumulative weight in a weighted TSA, the WCF of the transaction is also greater, and the transaction has a higher chance of being selected by the TSA.

The standard deviation (STD) TSA [9] aims to determine the α value to reduce the number of lazy and orphan tips in tangle. In a tangle with n transaction, parameter α in the STD algorithm varies depending on the cumulative weight of each tip among approvers:

$$\alpha = \frac{6}{n\sqrt{2}} STD \tag{3.28}$$

Where in equation 3.28 the value of α is changing based on the number of transactions (n) and the STD of the approvers weight.

As it is shown in figure 3.12, the main difference between the STD algorithm and weighted TSA is in the normalization of weights. Weights in a weighted TSA are normalized by subtracting the cumulative weight of each transaction from the cumulative weight

of the transaction with the highest cumulative weight.

$$W_{Normalised} = W - W_{max} \tag{3.29}$$

In the STD TSA, normalized weights are calculated by subtracting each transaction's cumulative weight from the transaction's mean cumulative weight:

$$W_{Normalised} = \sum_{i}^{n} \mid W - W_{mean} \mid \tag{3.30}$$

In a weighted TSA, the normalized weight of the transaction with the largest cumulative weight is zero ($W - W_{max} = 0$), which results in a WCF with a value of one. As a result, in a weighted TSA, the transaction with the highest cumulative weight has a high chance of being chosen.

The following example shows the impact of normalizing weights on choosing transactions in the weighted TSA:

Consider a group of approvers, that are the transactions with the cumulative weight of $[3,4,2,3,3,3]$ and the STD TSA is applied. Since the mean weight in $[3,4,2,3,3,3]$ is 3, the Squared Deviation (SD) is:

$$SD = \sum_{i=1}^{i=6} \mid W - W_{mean} \mid^2 = 2 \tag{3.31}$$

Given the SD of the approvers and the approvers length, equation 3.32 gives the variance of the approvers:

$$Variance = \frac{SD}{Numberof Approvers} = \frac{2}{6} = 0.33 \tag{3.32}$$

Therefore the STD of the approvers (equation 3.33) will be:

$$\sigma(STD) = \sqrt{0.33} = 0.57 \tag{3.33}$$

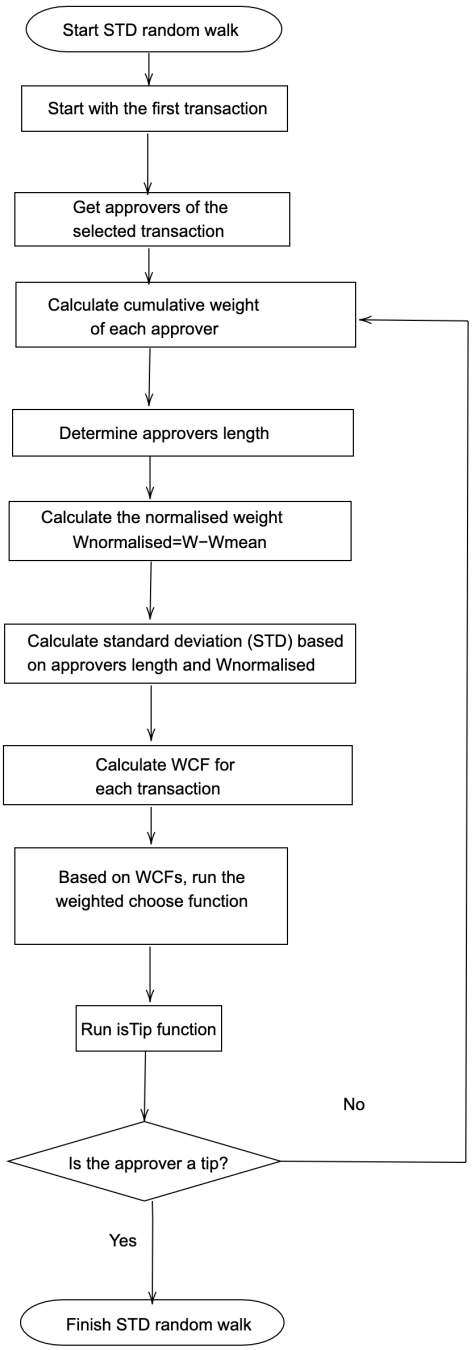


Figure 3.12: STD algorithm

With a total number of 500 transactions and equation 3.28, the value of α is 0.0048.

3.8 Ferraro's algorithm [14]

Ferraro's study [14] focuses on eliminating orphan tips in the DAG. The study offers a two-step random walk for each node to choose the tip. First, a node will run the TSA with a high α value to ensure that the transaction with the highest cumulative weight is chosen. The same node will then use a second TSA, but this time with a small value of α to ensure that the transaction with the lowest cumulative weight has the opportunity to be chosen by the node.

Assume we want to measure the approval probability of V_1 by a lazy tip in figure 3.13. If V_{19}, approves V_1 in Ferraro's algorithm, V_{19} is a lazy tip.

The first step in Ferraro's paper is referred to as the security step. In the security step, the tip TSA uses $\alpha = 1$. Figure 3.13 and equation 3.35 show the calculated WCFs for $[V_1, V_2, V_3]$:

$$\begin{pmatrix} w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 16 \\ 16 \end{pmatrix} \Rightarrow \begin{pmatrix} w_2 - w_{max} \\ w_3 - w_{max} \\ w_4 - w_{max} \end{pmatrix} = \begin{pmatrix} -15 \\ 0 \\ 0 \end{pmatrix} \tag{3.34}$$

$$\begin{pmatrix} WCF_2 \\ WCF_3 \\ WCF_4 \end{pmatrix} = \begin{pmatrix} e^{-\alpha * w_2} \\ e^{-\alpha * w_3} \\ e^{-\alpha * w_4} \end{pmatrix} = \begin{pmatrix} 3.05902321e-7 \\ 1 \\ 1 \end{pmatrix} \tag{3.35}$$

The total sum of the WCFs considering that WCF_4 is almost zero $(3.05902321e-7)$, is calculated as:

$$\sum WCF = WCF_2 + WCF_3 + WCF_4 = 1 + 1 + (3.05902321e-7) \simeq 2 \tag{3.36}$$

To check if the V_1 is selected by the TSA , the probability of V_2 is calculated:

$$Probability = \begin{pmatrix} \frac{WCF_2}{\sum WCF} \\ \frac{WCF_3}{\sum WCF} \\ \frac{WCF_4}{\sum WCF} \end{pmatrix} = \begin{pmatrix} 1.0196744e-7 \\ 0.50 \\ 0.50 \end{pmatrix} \tag{3.37}$$

Since the probability of V_2 being chosen by the TSA is $1.0196744e-7$, therefore V_2 is not selected by the security step in the TSA.

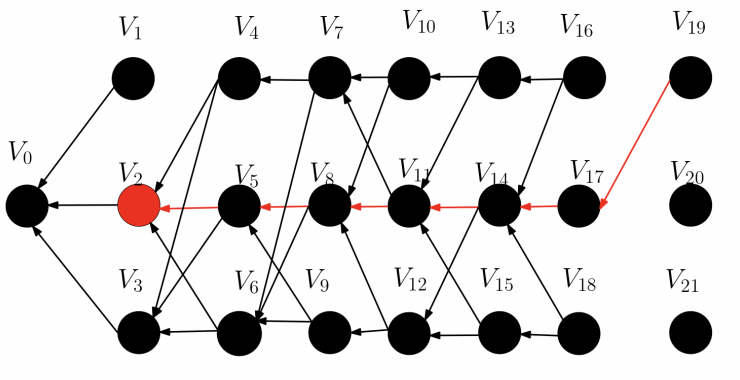


Figure 3.13: The chance of V_1 being selected by a lazy tip in the security step of the Ferraro's algorithm

In the second step (figure 3.14), the TSA uses $\alpha = 0.001$, known as the sweep step. The example below will also show the possibility of selecting an older transaction (V_1) in the sweepe step of Ferraro's TSA. The updated WCFs in figure 3.14 are:

$$\begin{pmatrix} w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 17 \\ 16 \end{pmatrix} \Rightarrow \begin{pmatrix} w_2 - w_{max} \\ w_3 - w_{max} \\ w_4 - w_{max} \end{pmatrix} = \begin{pmatrix} -16 \\ 0 \\ -1 \end{pmatrix} \tag{3.38}$$

since $\alpha = 0.1$ in the sweep step:

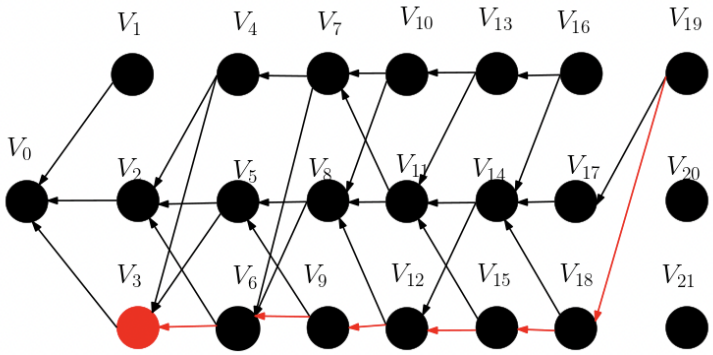


Figure 3.14: The chance of V_1 being selected by a lazy tip in the sweepe step of the Ferraro's algorithm

$$\begin{pmatrix} WCF_2 \\ WCF_3 \\ WCF_4 \end{pmatrix} = \begin{pmatrix} e^{-\alpha * w_2} \\ e^{-\alpha * w_3} \\ e^{-\alpha * w_4} \end{pmatrix} = \begin{pmatrix} 0.2 \\ 1 \\ 0.9 \end{pmatrix} \tag{3.39}$$

$$\sum WCF = WCF_2 + WCF_3 + WCF_4 = 0.2 + 1 + 0.9 \simeq 2.1 \tag{3.40}$$

The probability of each transaction being selected is:

$$Probability = \begin{pmatrix} \frac{WCF_2}{\sum WCF} \\ \frac{WCF_3}{\sum WCF} \\ \frac{WCF_4}{\sum WCF} \end{pmatrix} = \begin{pmatrix} 0.095 \\ 0.47 \\ 0.42 \end{pmatrix} \tag{3.41}$$

Therefore, V_1 is not selected by the Ferraro's TSA.

3.9 Hybrid TSA algorithm

In this section, we describe our proposed hybrid TSA. In the hybrid TSA algorithm, the α parameter is not a fixed value and changes with a random function. Therefore the WCF and the chance of selecting a tip is changing.

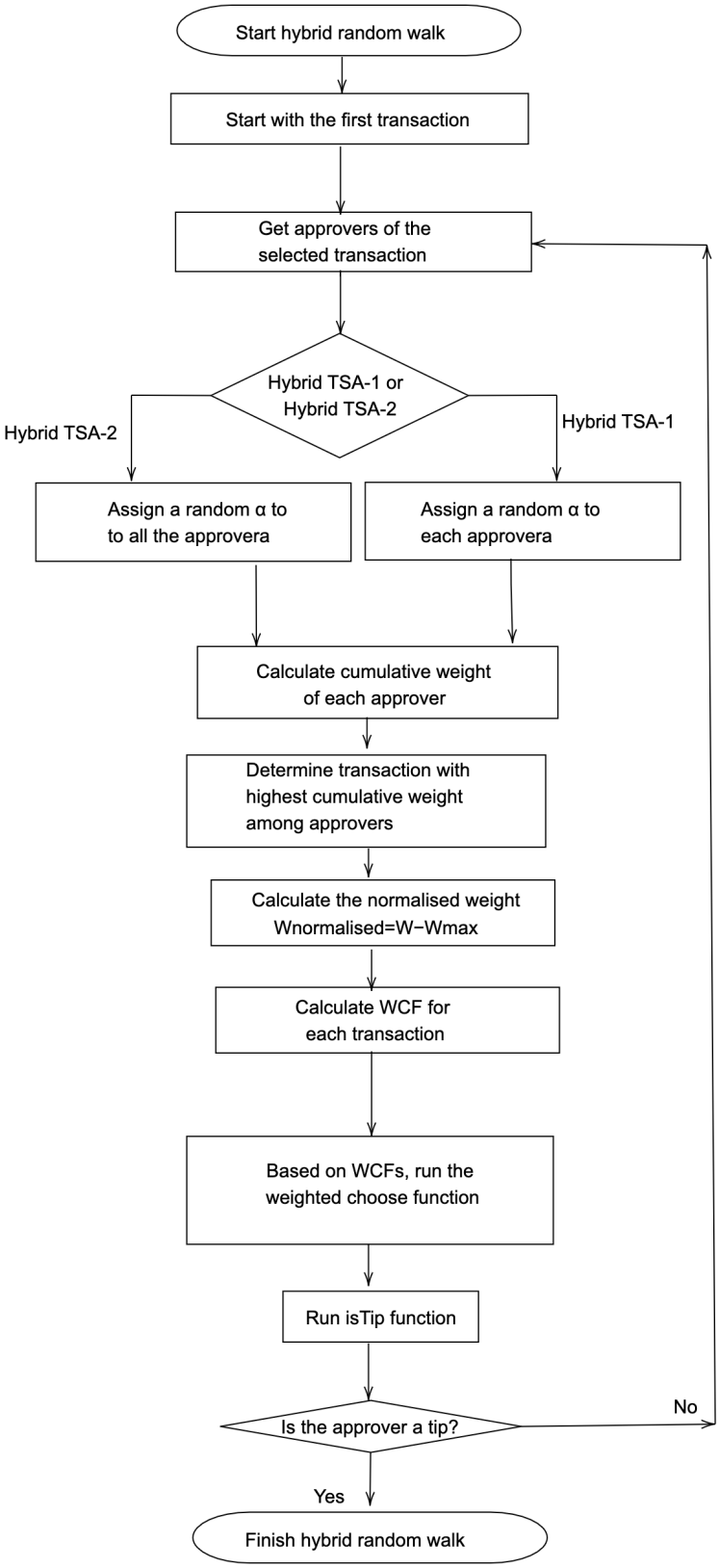


Figure 3.15: Hybrid TSA-1 and Hybrid TSA-2

In hybrid TSA-1, starting from the first transaction, the random walker finds the approver transactions (approvers function in the random walk section). The algorithm will then calculate the cumulative weight of each approver. Also, each approver will be assigned a random value of α. The random number generator in hybrid TSAs returns a floating-point, pseudo-random number in the range 0 to less than 1 (inclusive of 0, but not 1) with a uniform distribution over that range. The random value of α is different for each approver, and each time a node runs the TSA, approver transactions will get a new random α. Finally, the algorithm calculates the WCF for each approver, and if the approver is a tip, the hybrid random walker will stop. However, if the approver transaction is not a tip, the random walker continues looking for an approver that is a tip. Figure 3.15 shows the hybrid TSA-1 and hybrid TSA-2 method. The choose function in the hybrid TSA, is similar to the one we described in the weighted, STD and the Ferrero's TSA.

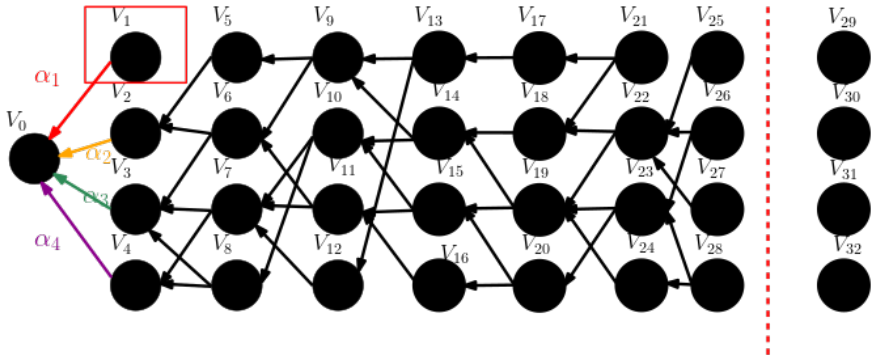


Figure 3.16: The chance of (V_1) being selected by hybrid TSA-1

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{pmatrix} 2 \\ 22 \\ 24 \\ 22 \end{pmatrix} \Rightarrow \begin{pmatrix} w_1 - w_{max} \\ w_2 - w_{max} \\ w_3 - w_{max} \\ w_4 - w_{max} \end{pmatrix} = \begin{pmatrix} -22 \\ -2 \\ 0 \\ -2 \end{pmatrix} \tag{3.42}$$

Suppose we have a sample of DAG including 29 transactions (V_0-V_{28}), where V_1 is the transaction waiting for its approval.The following example shows that the hybrid TSA-1

is less likely to select transaction V_1 as it is shown in figure (3.17). Suppose the random walker starts from V_0 in figure 3.17 and find the approvers for V_0 that are $[V_1, V_2, V_3, V_4]$. The TSA will then assign the random α_1, α_2, α_3, α_4 with a uniform distribution between $[0,1]$ to each approver transaction, in our example V_1, V_2, V_3, V_4, V_5. Suppose the random α values are:

$$[\alpha_1, \alpha_2, \alpha_3, \alpha_4] = [0.1, 0.8, 0.6, 0.4] \tag{3.43}$$

The WCF for each transaction is:

$$\begin{pmatrix} WCF_1 \\ WCF_2 \\ WCF_3 \\ WCF_4 \end{pmatrix} = \begin{pmatrix} e^{-\alpha * w_1} \\ e^{-\alpha * w_2} \\ e^{-\alpha * w_3} \\ e^{-\alpha * w_4} \end{pmatrix} = \begin{pmatrix} 0.11 \\ 0.2 \\ 1 \\ 0.44 \end{pmatrix} \tag{3.44}$$

The total cumulative weight is calculated as:

$$\sum WCF = WCF_2 + WCF_3 + WCF_4 = 0.11 + 0.2 + 1 + 0.44 = 1.75 \tag{3.45}$$

The probability of each transaction selected by the TSA is:

$$Probability = \begin{pmatrix} \frac{WCF_1}{\sum WCF} \\ \frac{WCF_2}{\sum WCF} \\ \frac{WCF_3}{\sum WCF} \\ \frac{WCF_4}{\sum WCF} \end{pmatrix} = \begin{pmatrix} 0.062 \\ 0.11 \\ 0.57 \\ 0.25 \end{pmatrix} \tag{3.46}$$

The example below shows the chance of an older tip being selected by a lazy tip in hybrid TSA-2. Suppose we have a similar sample of DAG mentioned in the previous example including 29 transactions (V_0-V_{28}), where V_1 is the transaction waiting for its approval. The

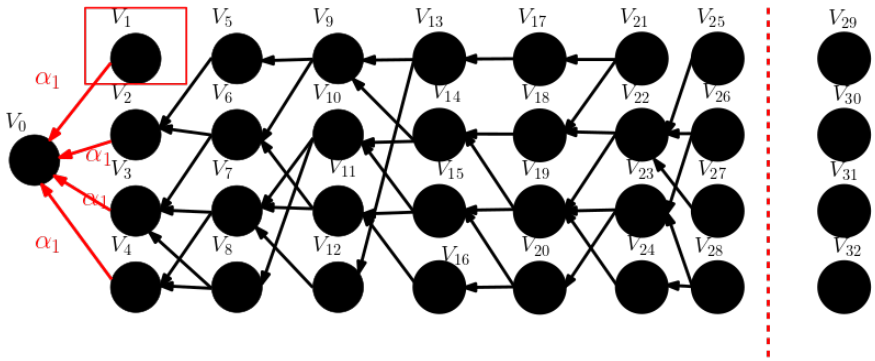


Figure 3.17: The chance that V_1 being selected by hybrid TSA-2

hybrid TSA-2 focus is to change α from one approver's array to another, meaning that α still will be the same among the approvers of a particular tip transaction, but α will change when the walker traverse to another transaction and finds new transaction's approvers. Figure 3.17 shows all the approver transactions of the genesis transaction V_0 (in our example V_1, V_2, V_3 and V_4) receive the same amount of α where α is the random number generated with the uniform distribution between $[0,1]$. Suppose the random α value generated is equal to 0.3, therefore, the WCF for the approvers (V_1, V_2, V_3 and V_4) is updated as:

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{pmatrix} 2 \\ 22 \\ 24 \\ 22 \end{pmatrix} \Rightarrow \begin{pmatrix} w_1 - w_{max} \\ w_2 - w_{max} \\ w_3 - w_{max} \\ w_4 - w_{max} \end{pmatrix} = \begin{pmatrix} -22 \\ -2 \\ 0 \\ -2 \end{pmatrix} \tag{3.47}$$

$$\begin{pmatrix} WCF_1 \\ WCF_2 \\ WCF_3 \\ WCF_4 \end{pmatrix} = \begin{pmatrix} e^{-\alpha * w_1} \\ e^{-\alpha * w_2} \\ e^{-\alpha * w_3} \\ e^{-\alpha * w_4} \end{pmatrix} = \begin{pmatrix} 0.0013 \\ 0.54 \\ 1 \\ 0.54 \end{pmatrix} \tag{3.48}$$

The total cumulative weight of each transaction is calculated as:

$$\sum WCF = WCF_1 + WCF_2 + WCF_3 + WCF_4 = 0.0013 + 0.54 + 1 + 0.54 = 2.0813 \quad (3.49)$$

The probability of each transaction selected by the TSA is:

$$Probability = \begin{pmatrix} \frac{WCF_1}{\sum WCF} \\ \frac{WCF_2}{\sum WCF} \\ \frac{WCF_3}{\sum WCF} \\ \frac{WCF_4}{\sum WCF} \end{pmatrix} = \begin{pmatrix} 0.00062 \\ 0.25 \\ 0.48 \\ 0.25 \end{pmatrix} \quad (3.50)$$

Therefore, V_1 becomes an orphan transaction.

Chapter 4

Discussion

4.1 Reviews and test results

In this chapter, we analyze the simulation result of different TSA algorithms. For a better examination of the simulated result, we use some of the parameters obtained from the IOTA tangle explorer. The IOTA tangle Explorer shows live IOTA transaction data and lets users search the IOTA tangle for transactions, addresses, bundles, and live market data. We start with the evaluation of TSA algorithms and how to identify lazy tips that occurred in the system as a result of TSA. We also discuss identifying orphan transactions in the tangle and examining the number of orphan transactions. We will finally introduce a metric to measure the efficiency of each TSA.

4.1.1 TSA evaluation

To evaluate the TSA's performance, we run the simulator with a set of parameters to answer the following questions:

- What is the average number of orphan tips with the given parameters (network types)?
- What is the average number of lazy tips as the result of TSA?

As we already discussed in the simulation section (chapter 3), the following is the transaction flow:

- Transaction creation: Transaction arrives randomly in the network. For each incoming transaction, a site is created. Each site contains the corresponding transaction and

its total weight.

- Tip selection: Each new site should approve two other unverified sites (tips).
- After the simulation run is completed, the simulator provides us with information such as the sites and the timing of their approval.
- Based on the site's approval timing, we calculate the number of orphan and lazy tip transactions.

4.1.2 Simulator output

Every transaction in the simulator has its unique name and timing, therefore, making it easy to identify orphan and lazy transactions. The simulator's output includes selected transactions (approved/child transactions) and the approver/parent transactions. The approver transactions includes two main objects:

- **Approver** transaction name (each transaction has a unique name)
- **Approver** transaction time (each transaction has a unique timing).

Also, transactions selected by the TSA provide us information about their approval. The approved transaction includes two main objects:

- **Approved** transaction name (each transaction has a unique name)
- **Approved** transaction time (each transaction has a unique timing)

The structure of the tangle's recorded output by the simulator is shown below:.

$$Transactions: \begin{cases} Approvers: & \begin{cases} TransactionName: 1,2,3,...,500 \\ TransactionTime: t=1, t=5, t=7, etc... \end{cases} \\ Approved: & \begin{cases} TransactionName: 0,...,499 \\ TransactionTime: t=0, t=6, t=7, etc... \end{cases} \end{cases} \tag{4.1}$$

4.1.3 Modifications we made into the simulator

We made the following changes to the simulator:

1. Counting number of orphan tips (sites with no approvals).

2. Defining lazy tips in the tangle, providing a threshold delay to identify lazy tips and counting the number of lazy tips for different values of time threshold.

In order to count the number of orphan transactions and the lazy tips we made some modifications to the simulator:

- Adding TSA files to the simulator: (STD TSA, Ferraro TSA and our proposed TSA hybrid TSA-1 and hybrid TSA-2)

- Adding a file to identify and count the number of orphan tips

- Adding a file to identify and count the number of lazy tips

Figure 4.1 shows the modification we made into the simulator (red areas). To count the number of orphan transactions, we wait until the last site runs the TSA, and then we look into the simulator output. We look into the sites with no parent transaction and count them as the orphan transactions.

4.1.4 Identifying orphan transactions in the simulator

Figure 4.2 shows the orphan transaction in a finite DAG. The gray transaction in simulator show tips with no parent transaction. Based on the orphan tip definition, V_8 and V_{15} are the orphan transactions. To count the number of orphan transaction in a bigger scale (500 transaction) in our simulation, the following steps are taken:

- We get all the transactions and their corresponding links from the simulator output.

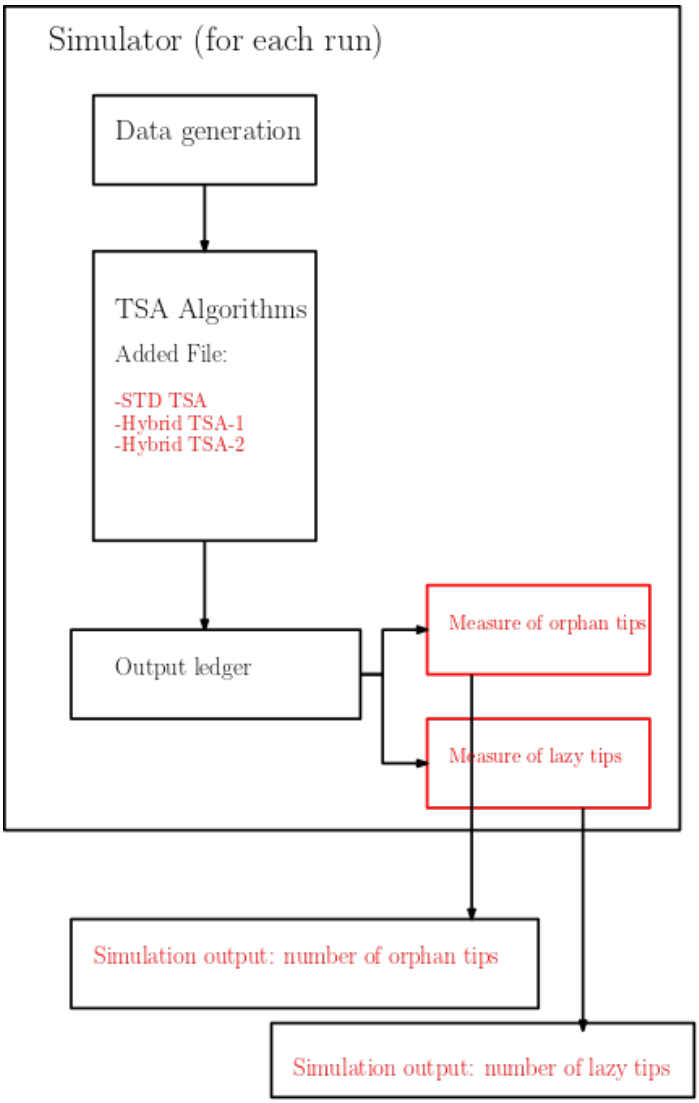


Figure 4.1: Modification we made into the original simulator

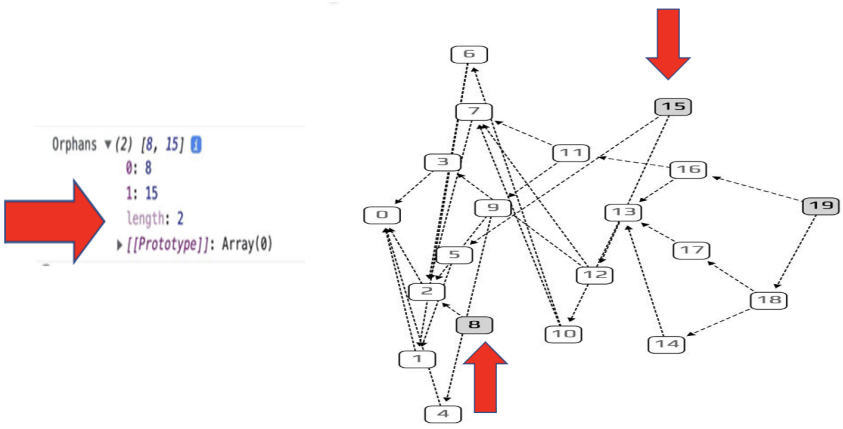


Figure 4.2: Orphan tips (the colour of transaction 19 is gray, which means transaction 19 has no parent, but only transaction 8 and 15 are orphan)

- Considering the **first** and the **last** transaction that gets **approval** ($[V_{first}, V_{last}]$, if any transaction is missing between the first and the last approved transaction, the missing transaction is orphan).

  In our example, $[V_0, V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_9, V_{10}, V_{11}, V_{12}, V_{13}, V_{14}, V_{16}, V_{17}, V_{18}, V_{19}]$, are the approved transactions. Since V_8 and V_{15} are missing transactions, we consider them as the orphan tips.

4.1.5 Identifying lazy tips as a result of TSA in the simulator

We decided to define different latency thresholds and determine which transactions are in the given range to measure the transactions with delays in their approvals. To decide whether a transaction is lazy, we do the following steps:

- We get all the transactions and their corresponding links until the last transaction. Every link includes data such as parent and child transactions and their timestamps.

- We set different time thresholds. Starting from a 2-second delay, we increase the sensitivity of delays to 9 sec. If the time difference between the child and parent transaction is larger than the threshold, the parent transaction is a lazy transaction.

Figure 4.3 shows an example of simulator output to identify emergence of lazy transactions as the result of the TSA. Transaction submitted at "1 : 53 : 00 pm" is a lazy transaction for the transaction occurring at "1 : 52 : 54 pm" (6-second approval) but is a normal parent for the transaction occurring at "1 : 52 : 58 pm" (2-second approval).

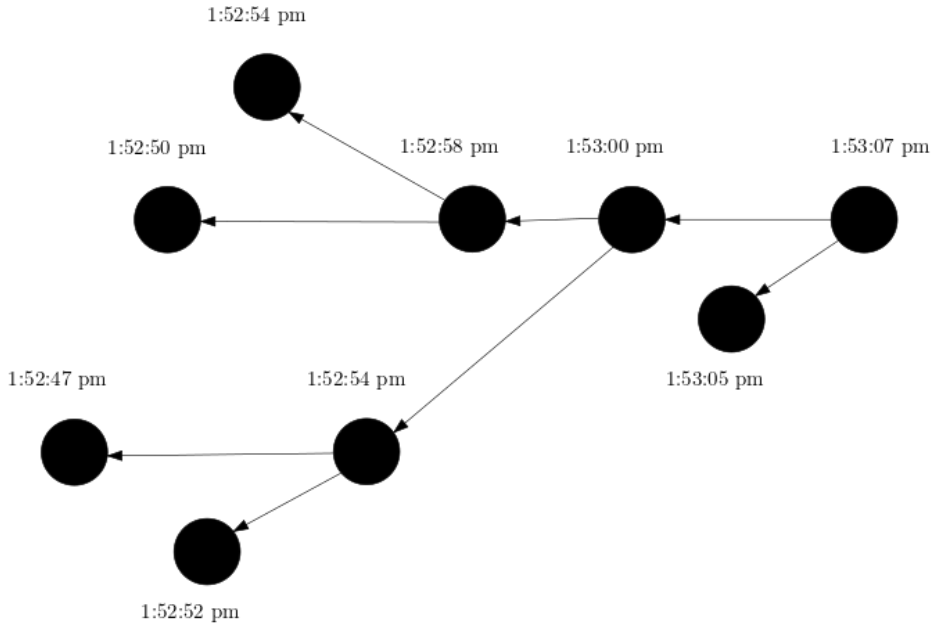


Figure 4.3: A sample of simulator's output

4.1.6 Emergence of lazy tips as a result of TSA in the real IOTA

Figure 4.4 shows our observation of transaction approvals in the real IOTA network by observing the timestamps of the approved transactions. In the IOTA network, the approval range of sequent transactions has a minimum of 1 sec (or less than that). We were only able to monitor a sample of transactions. We can conclude that based on the sample output data from the IOTA explorer the maximum approval range for a transaction is 9 seconds.

As we mentioned earlier, since the transaction's approval time is a stochastic process (T is not determined), we decided to use a range of approval time between two sequent transactions to verify our results .

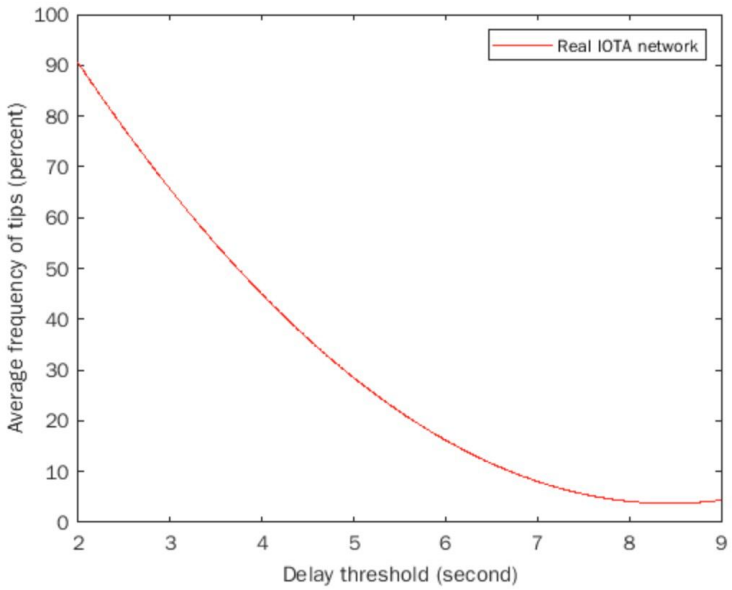


Figure 4.4: Lazy tips in real IOTA

4.1.7 Simulator parameters

We perform IOTA measurements for orphan and lazy tips in the previous IOTA legacy network (with the small number of λ) and the updated IOTA main-net (with a large value of λ). The transaction rate in the IOTA legacy network is around 0.5 , and in the best-case scenario, the arrival rate can reach up to 1. Tangle's transaction rate on the main net can reach up to 20 transactions per second (high-load), with an average of 15 transactions per second. Table 4.1, shows the transaction rate in different IOTA networks. The simulation is run with 500 nodes (500 submitted transaction). The hidden time is $h = 1$ and the value of λ changes depending on the network type (legacy, main net and high load main net). For each TSA, the simulation is run 10 times and the frequency of orphan tips are reported as the average number of orphan tips per 500 transaction.

4.1.8 Analyzing orphan transactions from the simulator's output

As the number of transactions on the main net grows, the weighted TSA performance degrades. The unweighted, uniform and STD algorithms have the fewest orphan tips in the legacy and main networks. On the other hand, the hybrid random walk surpasses the uni-

Table 4.1: Simulator parameters for the main-net and legacy network

Simulated Network	λ	h
Legacy net	0.5	1 Sec
Main-net	15	1 Sec
High load Main-net	20	1 Sec

form, unweighted, and STD algorithms in terms of orphans. The increased rate of orphan tips is due to eliminating more lazy tips from the system.

Figure 4.5 shows the histogram result of the measured orphan tips in different TSA's methods. The labels shown in figure 4.5 are the TSA algorithms explained in table 4.2.

Table 4.2: Labels we used in the simulator

TSA algorithm	discussion
Uniform TSA	Section 3.4: TSA uses uniform selection
Unweighted TSA	Section 3.5: TSA uses a random walk
Weighted TSA	Section 3.6: TSA uses a weighted random walk
STD TSA	Section 3.7: TSA uses a standard deviation of transaction's weights
Ferraro's TSA	Section 3.8: TSA uses $[0.1, 1]$ for the security and sweep step
Ferraro's TSA	Section 3.8: TSA uses $[0.2, 0.9]$ for the security and sweep step
Hybrid TSAs	Section 3.9: Our proposed TSAs by using random values of α

Table 4.3 shows the the average number of orphan transactions in network with different value of λ.

Identified lazy transactions in the uniform TSA

Multiple lazy tips are identified in the uniform TSA for a 3-second threshold. For a 9 second threshold in the network with $\lambda = 15$, 10 percent of total transactions were identified as lazy transactions. In the network with $\lambda = 20$ and a 5-second threshold, 52.2 percent of

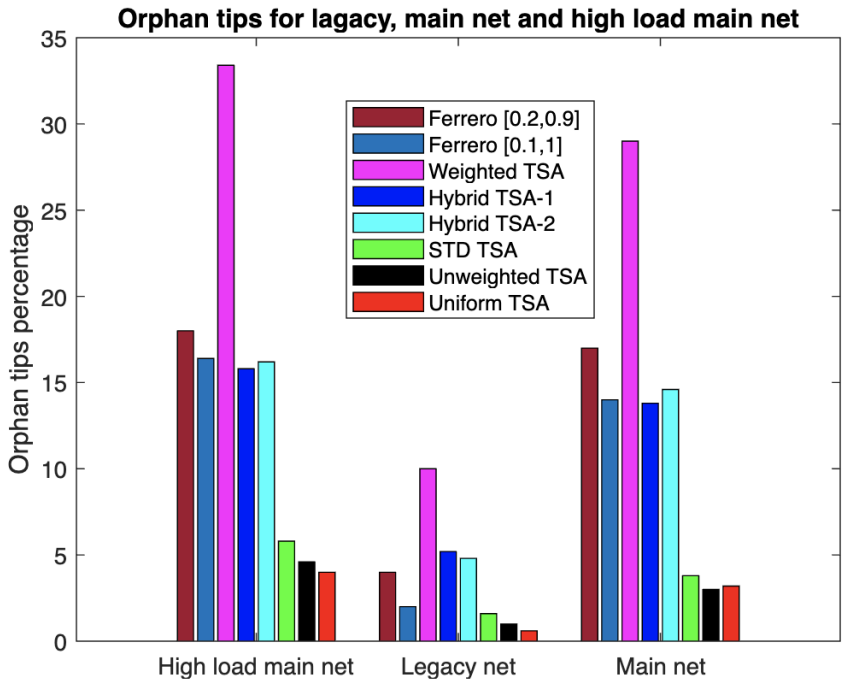


Figure 4.5: Orphan transactions in different TSA algorithms in legacy, main net and high load main net

Table 4.3: Orphan transactions in different TSAs

	Average number of orphan transactions		
TSA	λ=0.5	λ=15	λ=20
Uniform TSA	1	3.8	4
Unweighted TSA	1.6	3	4.6
Weighted TSA	10	30	33.4
STD TSA	3	4.4	5.8
Ferraro TSA $[0.2, 0.9]\}$	0.8	17	18
Ferraro TSA $[0.1, 1]\}$	0.4	14	16
Hybrid TSA-1	5	14.4	15.8
Hybrid TSA-2	5.2	15	16.2

total transactions were identified as lazy transactions. Also in a network with $\lambda = 20$, and a 3-second threshold, 59.4 percent of total transactions were identified as lazy transactions.

Table 4.4 shows the time threshold we used in the thesis and the colours we used to highlight different thresholds in networks with different value of λ:

Table 4.4: Colours we used for λ and $T_{Threshold}$

λ	$T_{Threshold}$	Colour	Discussion
$\lambda = 15$	9 Second		IOTA Suggested time threshold for IOTA 1.50 (Chrystal)
$\lambda = 20$	5 Second		Our Suggested time threshold for IOTA 2.00 (Coordicide.)
$\lambda = 20$	3 Second		IOTA Suggested time threshold for IOTA 2.00 (Coordicide)

Table 4.5 shows the average number of lazy transactions in the uniform TSA.

Table 4.5: Emergence of lazy tips as the result of uniform TSA

		Lazy tips percentage		
TSA	Threshold (Sec)	λ=0.5	λ=15	λ=20
Uniform TSA	9	0.8	10	20.2
	8	2	19.4	33.6
	7	3.8	26.8	43
	6	8	31.6	49.4
	5	16.4	34.6	52.2
	4	23.2	40	56.4
	3	32.8	46.6	59.4
	2	36	57.6	64.6

Identified lazy transactions in the unweighted TSA

When the unweighted TSA is run in the network with $\lambda = 0.5$, 32 percent of total transactions were identified as lazy transactions. Lazy transactions grow in networks with $\lambda = 15$ and $\lambda = 20$, resulting in 60.6 percent and 62.4 percent of all transactions, respectively. Fewer lazy tips were identified in the unweighted TSA for a 9-second threshold

accounting for 1% of the total transactions. Also, for a 9-second threshold in the network with $\lambda = 15$ and $\lambda = 20$, the number of lazy transactions are 9% and 16.4% of total transactions, respectively. The result of lazy tips in the unweighted TSA is shown in table 4.6:

Table 4.6: Emergence of lazy tips as the result of unweighted TSA

		Lazy tips percentage		
TSA	Threshold (Sec)	λ=0.5	λ=15	λ=20
Unweighted TSA	9	1	9	16.4
	8	1.6	15.2	29
	7	3.2	22.6	37.2
	6	8.8	28.6	43.8
	5	15.6	34	50.8
	4	25	37.8	56.6
	3	28.4	45.2	59.6
	2	32	60.6	62.4

Identified lazy transactions in the weighted TSA

In the weighted TSA, the number of identified lazy tips is significantly reduced. There are no lazy tips for a 9-second threshold in the networks with $\lambda = 15$. For a 3-second threshold, 6.4 percent of total transactions are lazy in a network with $\lambda = 20$. For our proposed 5-second threshold, 2.8 percent of total transactions are lazy in a network with $\lambda = 20$.

Table 4.7 shows the lazy tips percentage in different TSA algorithms.

Identified lazy transactions in STD TSA

In the STD TSA, the number of lazy tips is lower than in the uniform and unweighted TSAs, however, the number of lazy transactions is still larger than in the weighted TSA. In the network with $\lambda = 0.5$ and a 9-second threshold, the number of lazy tips accounts for 1.2 of the total transactions. In the same threshold, the lazy transactions increase to 3.6 and 14.4. For a 3-sec threshold in the network with $\lambda = 20$, the lazy transactions result in

Table 4.7: Emergence of lazy tips as the result of weighted TSA

		Lazy tips percentage		
TSA	Threshold(Sec)	λ=0.5	λ=15	λ=20
Weighted TSA	9	0	0	0
	8	0	0	0
	7	0	0.4	1
	6	0	1.2	1.8
	5	0.4	3.2	2.8
	4	1	3.6	5
	3	2.4	5.2	6.4
	2	3.7	6.6	10

46 percent of total transactions, whereas in the network with $\lambda = 20$ and a 5-sec threshold, the number of lazy transactions is 34.2 percent of total transactions respectively. Table 4.8 shows the result of lazy tips in the STD TSA.

Table 4.8: Emergence of lazy tips as the result of STD TSA

		Lazy tips percentage		
TSA	Threshold (Sec)	λ=0.5	λ=15	λ=20
STD TSA	9	1.2	3.6	14.4
	8	2	6.6	18.4
	7	3	10.4	24.2
	6	3.8	15	27.8
	5	6.4	20.4	34.2
	4	12.4	24.8	39.2
	3	31.4	20	46.6
	2	18	44.2	55.4

Identified lazy transactions in Ferraro TSA

In Ferraro's TSA, we simulate 2 different values. We first simulated a network with $\alpha = 0.1$ as the sweep step, and $\alpha = 1$ as the security step (we call it Ferraro $[0.1, 1]$). We also simulated a network with $\alpha = 0.2$ as the sweep step, and $\alpha = 0.9$ as the security step

(we call it Ferraro $[0.2, 0.9]$). In a network with $\lambda = 0.5$, $\lambda = 15$ and $\lambda = 20$, Ferraro $[0.1, 1]$ identifies lazy transactions accounting as the maximum of 1 percent of total transactions. In the network with $\lambda = 0.5$, 36 percent of total transactions are lazy. 70.2 and 77.8 percentage of total transactions are lazy in a network with $\lambda = 15$ and $\lambda = 20$.

In Ferrero $[0.2, 0.9]$, no lazy transaction appeared as the result of TSA after a 9-second threshold in the network with $\lambda = 15$. In a 3-sec threshold, the number of lazy transactions is 20 percent of the total transactions in a network with $\lambda = 20$. Also, In a 5-sec threshold, the number of lazy transactions is 5 percent of the total transactions in a network with $\lambda = 20$.

Table 4.9: Emergence of lazy tips as the result of Ferraro$[0.1, 1]$ TSA

		Lazy tips percentage		
TSA	Threshold (Sec)	λ=0.5	λ=15	λ=20
Ferraro TSA $[0.1, 1]$	9	1	0.5	0
	8	3	1	0
	7	5.6	3.8	1.8
	6	6.4	4.5	2.6
	5	10.2	8.2	5
	4	23.8	28.9	10
	3	58	53	20
	2	36	70.2	77.8

Identified lazy transactions in hybrid TSA

Hybrid TSA-1 and TSA-2 have the minimum amount of lazy tips among all the TSA algorithms. For a 9-second threshold, hybrid TSA-1 has the maximum of 0.8 percent of the total lazy transactions. In a 3-second threshold, the maximum percentage of lazy transactions is 9 percent of the total transactions. Also, for a 5-second threshold in a network with $\lambda = 20$, the maximum percentage of lazy transactions is 5.8 percent of the total transactions. for a Table 4.11 and 4.12 show the result of lazy transactions in the hybrid TSAs.

Table 4.10: Emergence of lazy tips as the result of Ferraro$[0.2, 0.9]$ TSA

		Lazy tips percentage		
TSA	Threshold (Sec)	λ=0.5	λ=15	λ=20
Ferraro TSA $[0.2, 0.9]$	9	0	0	0
	8	0	0	1
	7	0	1.8	3
	6	0	2.6	6
	5	1.2	5	8
	4	3.6	10	12
	3	10	20	25
	2	28.2	35	39

Table 4.11: Emergence of lazy tips as the result of hybrid TSA-1

		Lazy tips percentage		
TSA	Threshold (Sec)	λ=0.5	λ=15	λ=20
Hybrid TSA-1	9	0	0.8	1.6
	8	0	1.4	2.4
	7	0	3.6	3.8
	6	0.6	5.4	4.2
	5	1	6.2	5.8
	4	2.4	7.8	8.4
	3	3.6	9	9
	2	7	11.4	13.2

Overview of TSA algorithms: Which TSA is better?

As the delay range increases, we expect to have lazy tips, in which their corresponding node approves an old transaction (8 seconds or more). For the legacy net, the hybrid TSAs almost have no lazy tips. The hybrid TSAs have the least lazy tips after an 8-second delay with a growing range of transactions in the main net. The uniform, unweighted, and the STD TSA have the highest number of lazy tips in 8-sec or more delay. The weighted TSA is also experiencing no lazy tips both in the main net and the legacy net.

Given that the maximum approval time is 9 seconds, the uniform, unweighted and STD

Table 4.12: Emergence of lazy tips as the result of hybrid TSA-2

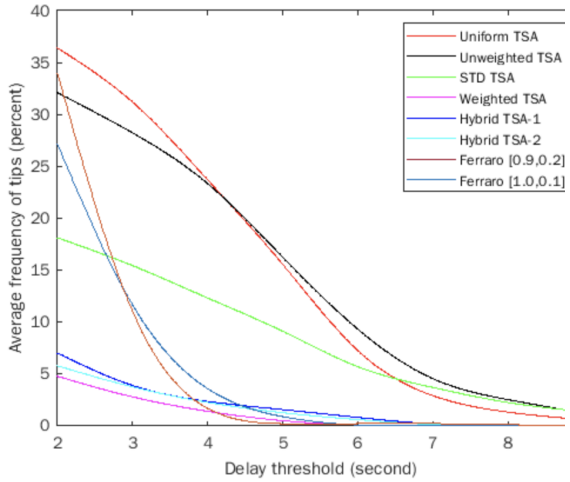
		Lazy tips percentage		
TSA	Threshold (Sec)	λ=0.5	λ=15	λ=20
Hybrid TSA-2	9	0	0.6	0.6
	8	0	1.6	1.6
	7	0	3.2	1.8
	6	0.2	4.2	3
	5	0.4	5.6	4.4
	4	2	7	8
	3	4	8.4	9.6
	2	7.6	10.6	13.2

TSA have the least resilience against lazy tips. Ferraro's TSA is also having a large number of lazy tips in a 2-second threshold. In the main-net simulation, the uniform, unweighted, and STD TSA will have the largest number of lazy tips. If the main-net simulator is expe-riencing high load transaction arrival ($\lambda = 20$), the frequency of lazy tips in the unweighted and uniform TSA is more than 60 percent of total tips and 80 percent for the Ferraro's TSA. The weighted and hybrid TSAs are slower than the uniform algorithm, but they can elimi-nate lazy tips. If the nodes approve a transaction with latency, the submitted transaction by the lazy nodes will remain unconfirmed. For a 2 second delay on the main net, the hybrid TSAs leave around 12 percent lazy tips. As latency grows to 9 seconds, lazy tips reduce to 0.8 percent lazy tips, proving that hybrid TSA-2 is resilient to lazy nodes in the main network.

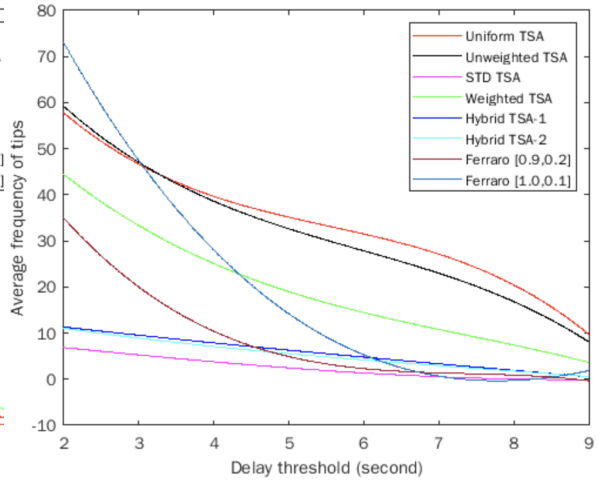
To measure the quality of transactions, we introduce an quality metric to show what percentage of total lazy transactions are orphans:

$$Quality = \frac{OrphanTransactions}{LazyTransactions} \tag{4.2}$$

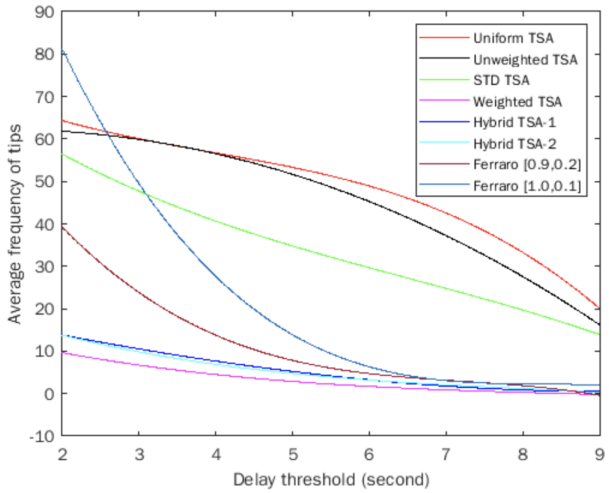
When the quality metric is 1, it will indicate that most of the lazy transactions that have been appeared in the tangle as the result of TSA, will remain orphans. Taking a look at the result of uniform and the unweighted TSA, we can see that the quality measure is



(a) λ=0.5



(b) λ=15



(c) λ=20

Figure 4.6: Lazy tips in different TSA algorithms. λ.

near zero, which means that although the number of lazy tips is the largest in the uniform and unweighted TSA, almost no orphan transactions exist in the tangle, and all the lazy transactions are approved in the tangle. On the other hand, the quality measure for the weighted TSA is 2.7 for the network with $\lambda = 0.5$ and increases to 4.54 for a network with $\lambda = 15$ meaning that not only lazy transactions remained orphan, but also some of the transactions that were not lazy remained orphan. Ferraro TSA $[0.2, 0.9]$ shows better quality in a network with $\lambda = 15$ and $\lambda = 20$ with an quality equal to 0.5) showing that

lazy transactions remained orphans in the tangle. However, in Ferrero's TSA, the other half of lazy tips will receive approvals in the tangle. The hybrid TSA-1 shows the best quality among all the other TSA algorithms with an quality equal to 1.19 meaning that the majority of the lazy transactions didn't receive approvals and remained orphans. Table 4.13 shows the result of the quality metric on different TSAs:

Table 4.13: Quality of transactions in different TSA algorithms

	Quality		
TSA	λ=0.5	λ=15	λ=20
Uniform TSA	0.027	0.065	0.061
Unweighted TSA	0.05	0.049	0.073
Weighted TSA	2.70	4.54	3.34
STD TSA	0.16	0.099	0.104
Ferraro TSA $[0.2, 0.9]\}$	0.028	0.48	0.46
Ferraro TSA $[0.1, 1]\}$	0.01	0.19	0.20
Hybrid TSA-1	0.71	1.26	1.19
Hybrid TSA-2	0.68	1.41	1.22

When comparing the results of different TSAs, the quality measurement is the most useful. For example, in a network with $lambda = 20$, we want to compare the results of the hybrid and STD TSAs.

As shown in the result of STD TSA in table 4.3, for a 5-second threshold, 5.8 percent of total transactions are considered orphan tips, whereas 34.2 percent of total transactions are considered lazy transactions (table 4.8).

On the other hand, in the hybrid TSA, around 15.8 percent of total transactions are considered orphan transactions (table 4.3) whereas 5.8 percent of transactions are considered lazy (table 4.11). It seems that the hybrid TSA is losing more transactions and leaving more transactions as orphans, however, when examining the quality measurement, the STD TSA received a 0.1 score, while the quality measure for the hybrid TSA is 1.19, indicating that

the hybrid TSA includes more high-quality transactions.

Chapter 5

Conclusion and Future Work

5.1 Summary

In this thesis, we studied tangle's different features, such as orphan and lazy tips and the cause they are created in tangle. We started the thesis by reviewing the literature on distributed ledger applications such as blockchain and DAG data structure. We then studied the tangle simulation using the tangle visualization tool. Throughout the simulation, we analyzed the result of orphan and lazy tips and how they can rely on the transaction rate. We leveraged the simulator to design the new hybrid TSA algorithms based on the adaptive value of the random walks in the algorithm. The algorithm efficiency is determined by measuring the orphan and the lazy tips in tangle under different tangle parameters and latency of the nodes.

Through the simulation, we analyzed the impacts of different parameters, including node's latency, the random walker adaptive behaviour and the transaction rate in the tangle, both for the previous network (legacy-net) and the recent network (main-net).

In the hybrid TSAs, tip transactions in tangle will not get a chance to be approved by the lazy node. Also, the left behind tips are less likely to be approved by the lazy tips, and fewer transactions are left unvalidated (orphan).

The current measures of the tangle can be extended for further studies on other characteristics of the tangle network. The study of the tip selection features can provide insight into tangle's performance in approving transactions and improve tangle's scalability by speeding up the tangle performance. Following is the future research direction:

1. Examining different distribution probabilities to measure the chance of transactions with different WCFs being selected by the TSA.

2. Instead of employing the weight coefficient factor on the weighted walker to balance lazy and orphan tips, a machine learning application may be used to categorize the lazy and orphan tips on the random walker.

Bibliography

[1] IOTA 1.50 Chrysalis chrysalis release. https://chrysalis.iota.org/. Accessed: 2021-09-30.

[2] IOTA 2.00 DEVNET nectar release. https://v2.iota.org/. Accessed: 2021-09-30.

[3] Gol Alon. Iota visualization library. https://github.com/iotaledger/iotavisualization.git, 2018.

[4] Gol Alon. Iota repository proposal. https://iotaledger.github.io/tips/, 2022.

[5] Vidal Attias and Quentin Bramas. Tangle analysis for iota cryptocurrency. 2018.

[6] Mohan Bhandary, Manish Parmar, and Dayanand Ambawade. A blockchain solution based on directed acyclic graph for iot data security using iota tangle. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pages 827–832. IEEE, 2020.

[7] Leroy Brice. Jstat. https://github.com/jstat/jstat.git, 2016.

[8] Gewu Bu, Önder Gürcan, and Maria Potop-Butucaru. G-iota: Fair and confidence aware tangle. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 644–649. IEEE, 2019.

[9] Fatemeh Sedighipour Chafjiri and Mohamad Mehdi Esnaashari Esfahani. An adaptive random walk algorithm for selecting tips in the tangle. In *2019 5th International Conference on Web Research (ICWR)*, pages 161–166. IEEE, 2019.

[10] Anamika Chauhan, Om Prakash Malviya, Madhav Verma, and Tejinder Singh Mor. Blockchain and scalability. In *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 122–128. IEEE, 2018.

[11] Janvi Dattani and Harsh Sheth. Overview of blockchain technology. *Asian Journal of Convergence in Technology*, 5(1):1–3, 2019.

[12] Advait Deshpande, Katherine Stewart, Louise Lepetit, and Salil Gunashekar. Distributed ledger technologies/blockchain: Challenges, opportunities and the prospects for standards. *Overview report The British Standards Institution (BSI)*, 40:40, 2017.

[13] Pietro Ferraro and Daniel Conway. Distributed ledger technologies and the collaborative economy. In *Analytics for the Sharing Economy: Mathematics, Engineering and Business Perspectives*, pages 95–107. Springer, 2020.

[14] Pietro Ferraro, Christopher King, and Robert Shorten. Iota-based directed acyclic graphs without orphans. *arXiv preprint arXiv:1901.07302*, 2018.

[15] IOTA Foundation. IOTA.org: explaining mana in iota, 2020.

[16] I Gusti Ayu Kusdiah Gemeliarana and Riri Fitri Sari. Evaluation of proof of work (pow) blockchains security network on selfish mining. In *2018 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, pages 126–130. IEEE, 2018.

[17] Johannes Göbel, Holger Paul Keeler, Anthony E Krzesinski, and Peter G Taylor. Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. *Performance Evaluation*, 104:23–41, 2016.

[18] Li-Yuan Hou, Tsung-Yi Tang, and Tyng-Yeu Liang. Iota-bt: A p2p file-sharing system based on iota. *Electronics*, 9(10):1610, 2020.

[19] Evgeniy O Kiktenko, Nikolay O Pozhar, Maxim N Anufriev, Anton S Trushechkin, Ruslan R Yunusov, Yuri V Kurochkin, AI Lvovsky, and Aleksey K Fedorov. Quantum-secured blockchain. *Quantum Science and Technology*, 3(3):035004, 2018.

[20] ID Kotilevets, IA Ivanova, IO Romanov, SG Magomedov, VV Nikonov, and SA Pavelev. Implementation of directed acyclic graph in blockchain network to improve security and speed of transactions. *IFAC-PapersOnLine*, 51(30):693–696, 2018.

[21] Jari Kreku, Visa Antero Vallivaara, Kimmo Halunen, Jani Suomalainen, M Ramachandran, V Muñoz, V Kantere, G Wills, and R Walters. Evaluating the efficiency of blockchains in iot with simulations. In *IoTBDS*, pages 216–223, 2017.

[22] Bartosz Kusmierz, William Sanders, Andreas Penzkofer, Angelo Capossele, and Alon Gal. Properties of the tangle for uniform random and random walk tip selection. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 228–236. IEEE, 2019.

[23] Bartosz Kusmierz, Philip Staupe, and Alon Gal. Extracting tangle properties in continuous time via large-scale simulations. *Technical Report. working paper*, pages 2018–08, 2018.

[24] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. In *Concurrency: the Works of Leslie Lamport*, pages 203–226. 2019.

[25] Mohamed Riswan Abdul Lathif, Pezhman Nasirifard, and Hans-Arno Jacobsen. Cidds: A configurable and distributed dag-based distributed ledger simulation framework. In *Proceedings of the 19th International Middleware Conference (Posters)*, pages 7–8, 2018.

[26] Author Dr. Liew. Tangle archives, Oct 2018.

[27] Faraz Masood and Arman Rasool Faridi. Consensus algorithms in distributed ledger technology for open environment. In *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, pages 1–6. IEEE, 2018.

[28] Faraz Masood and Arman Rasool Faridi. An overview of distributed ledger technology and its applications. *Int. J. Comput. Sci. Eng*, 6(10):422–427, 2018.

[29] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[30] Satoshi Nakamoto and A Bitcoin. A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, 4, 2008.

[31] Michael Nofer, Peter Gomber, Oliver Hinz, and Dirk Schiereck. Blockchain. *Business & Information Systems Engineering*, 59(3):183–187, 2017.

[32] Serguei Popov. The tangle. *cit. on*, page 131, 2016.

[33] Serguei Popov. The tangle. *White paper*, 1(3), 2018.

[34] Serguei Popov and Q Lu. Iota: feeless and free. *IEEE Blockchain Technical Briefs*, 2019.

[35] Serguei Popov, Hans Moog, Darcy Camargo, Angelo Capossele, Vassil Dimitrov, Alon Gal, Andrew Greve, Bartosz Kusmierz, Sebastian Mueller, Andreas Penzkofer, et al. The coordicide. *Accessed Jan*, pages 1–30, 2020.

[36] Serguei Popov, Olivia Saa, and Paulo Finardi. Equilibria in the tangle. *Computers & Industrial Engineering*, 136:160–172, 2019.

[37] Wei Ren, Jingjing Hu, Tianqing Zhu, Yi Ren, and Kim-Kwang Raymond Choo. A flexible method to defend against computationally resourceful miners in blockchain proof of work. *Information Sciences*, 507:161–171, 2020.

[38] Sonmez Sinan. React. `https://github.com/facebook/react.git`, 2011.

[39] Iain Stewart, Daniel Ilie, Alexei Zamyatin, Sam Werner, MF Torshizi, and William J Knottenbelt. Committing to quantum resistance: a slow defence for bitcoin against a fast quantum computing attack. *Royal Society open science*, 5(6):180410, 2018.

[40] Lyubomir Stoykov, Kaiwen Zhang, and Hans-Arno Jacobsen. Vibes: fast blockchain simulations for large-scale peer-to-peer networks. In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Posters and Demos*, pages 19–20, 2017.

[41] Ali Sunyaev. Distributed ledger technology. In *Internet Computing*, pages 265–299. Springer, 2020.

[42] Luming Wan, David Eyers, and Haibo Zhang. Evaluating the impact of network latency on the safety of blockchain transactions. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 194–201. IEEE, 2019.

[43] Tengfei Xue, Yuyu Yuan, Zahir Ahmed, Krishna Moniz, Ganyuan Cao, and Cong Wang. Proof of contribution: A modification of proof of work to increase mining efficiency. In *2018 IEEE 42nd annual computer software and applications conference (COMPSAC)*, volume 1, pages 636–644. IEEE, 2018.

[44] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE international congress on big data (BigData congress)*, pages 557–564. IEEE, 2017.

[45] N Živi, E Kadušić, and K Kadušić. Directed acyclic graph as tangle: An iot alternative to blockchains. In *2019 27th Telecommunications Forum (TELFOR)*, pages 1–3. IEEE, 2019.