How to Choose Its Parents in the Tangle

Vidal Attias, Quentin Bramas

How to Choose its Parents in the Tangle

Vidal Attias[1] and Quentin Bramas[2]

[1] ENS Rennes
[2] ICUBE, Strasbourg University, CNRS

Abstract. The Tangle is a data structure mainly used to store transactions in the IOTA cryptocurrency. It has similarities with the blockchain structure of Bitcoin but in the Tangle, a block contains only one transaction and has not one, but two parents. The security and the stability of this distributed data structure is highly dependent on the algorithm used to select the parents of a new block.
Previous work showed that the current parents selection algorithms are insecure, not stable or have low performances. And we propose a new algorithm that combines all these properties.

Keywords: Blockchain · Distributed ledger · Security · Performances evaluation

1 Introduction and Model

A Distributed Ledger Technology (DLT) is (*i*) an append-only data structure and (*ii*) a protocol that defines how the agents in a network agree to append new data. The Bitcoin is the most famous example. It uses the blockchain to store the transactions and the Proof of Work (PoW) to elect a node in the network responsible for writing a new block. In this paper, we are interested in the data structure called *Tangle*, used to store transactions in the IOTA cryptocurrency, and especially in the algorithm used to append new data. We present a new algorithm and show it offers better performances compared to existing algorithms.

The Tangle We consider a set of connected agents generating transactions. The transactions are stored in a Directed Acyclic Graph (DAG) called *Tangle*. Each agent has a local copy of the Tangle. A vertex of the Tangle, called *site*, represents a transaction and has two parents (potentially the same one) in the Tangle. A site is said to *directly confirm* its parents and *indirectly confirm* its other ancestors in the Tangle. A *tip* of the Tangle is a site which has no child, i.e which isn't confirmed by any site. The *genesis* is the only site with no parents.

In order to include a transaction in the Tangle, an agent must perform a proof of work, i.e, solving a cryptographic puzzle requiring a certain amount of computational power. The *weight* of a site represents this work and we assume each site has a weight of 1. Then, the *cumulative weight* of a site is defined [Pop16] as the sum of its own weight with the weight of its descendants (the sites which confirm it). See Figure 1 for an illustration.

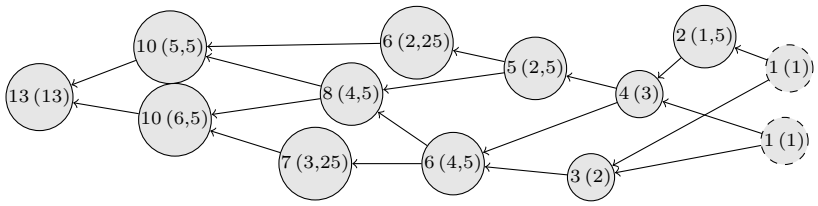

Fig. 1: An example of a Tangle. In each site is writen its cumulative weight and in bracket its real cumulative weight (see Section 2).

Tips Selection Algorithm (TSA) When a site is added to the Tangle, its parents are selected by a *Tip Selection Algorithm* (TSA). The TSA must select two tips (unconfirmed sites) that are not conflicting. The TSA is a fundamental component of the protocol because it implicitly indicates what is the current view of the agent generating the new site. Indeed, if two tips are conflicting, the TSA indicates which one is considered correct (and should be extended by appending a new site to it) or orphaned (by ignoring it).

Each site in the Tangle has two parents but a site can have multiple children, because of the local versions of the Tangle can be different for two agents in the network, due to the latency. The TSA could chose a site which is a tip in the local version on the Tangle, but that is already confirmed in another one.

The whitepaper [Pop16] presents two TSA[3]:

- Uniform TSA: Each parent is chosen uniformly at random among all the tips.
- Markov Chain Monte Carlo (MCMC): the selection of each parent is done by using a random walk. A walker starts from a given site (eg, the genesis), moves from site to child site, and stops when it reaches a tip. At each site, it uses a transition function depending of the site's cumulative weight denoted by w. In a site v, the probability of going to a child $u \in C(v)$ is:

$$p_{v,u} = \exp(-\alpha(w(v) - w(u)))/ \sum_{c \in C(v)} \exp\left(-\alpha(w(v) - w(c))\right)$$

  where $\alpha > 0$ is a parameter of the algorithm. This TSA is currently used in production in the IOTA cryptocurrency with the parameter $\alpha = 0.001$. We denote this TSA MCMC($\alpha = 0.001$) in the remainder. One may notice that as we send two random walks, we must enforce that the two tips are not in conflict.

The Double Spending Problem The double spending attack in a DLT consists in generating two transactions using the same funding source but with two distinct recipients. The aim for the attacker is to persuade each recipient that the transaction it receives is valid and that the other one is not.

[3] A third one is briefly presented but is actually just a variation of the MCMC that we present here.

In order to simplify, one recipient can be a car seller and the second one the attacker himself. The attacker broadcasts the first transaction and then waits for the car seller to be convinced of the validity of its transaction and to give the car's keys. Then, the attacker broadcasts the second transaction, and generates multiple other transactions to make the first one invalid. When all the network's agents consider the first transaction as invalid, the seller will have given its car without having got the money.

To prevent a double spend attack in the Tangle, it is necessary that computational power used by the honest agents to generate their sites[4] is greater than the computational power of the adversary [Bra18].

Contributions In this article, we presents a new way to compute the cumulative weight and a new TSA using this new weight. Then, we analyse theoretically and with simulations our new TSA and show that it improves the MCMC currently used in production by IOTA on some points, while being as good on other points : (i) it gets executed much more quickly, which is a real advantage for the network's nodes generating a great number of transactions, (ii) it doesn't leave unconfirmed transactions, even in high load and (iii) it is secure.

2 A New Way to Compute the Cumulative Weight

The main issue of the cumulative weight as defined previously is that for a given site, its children's cumulative weight does not give any indications on the real weight of the subtangles 4confirming them. Let us consider a site with two children, each having a cumulative weight of 10. Then, there is no way to differentiate between the case (a) where 9 sites are confirming both children and the case (b) where 9 sites are confirming the first child and 9 other sites are confirming the second child. In the first case, the corresponding cumulative weight should be lower since it is easier to generate the 11 sites in the first case compared to the 20 sites in the second one. When the cumulative weight is used in a random walk, the probability to move toward a subtangle should not depend on the number of children connected to this subtangle.

We define the *real cumulative weight* $R(u)$ of a site u as one plus half the sum of its children's real cumulative weights:

$$R(u) = 1 + \sum_{c \in C(u)} R(c)/2 \qquad (1) \qquad\qquad p_{v,u} = R(u)/\sum_{c \in C(v)} R(c) \qquad (2)$$

This makes sense because half of each site's weight contributes to each of its parent[5]. Also, one can notice that the sum of real cumulative weights of any subset of children correspond exactly to the amount of power used to confirm those children. We also decided to change the formula used to compute probabilities

[4] Observe here that the computational power *used* by the honest agents is not the same as the computational power of the honest agent. Indeed, the former could be much lower than the latter.

[5] This can easily be generalized to the case where each site has k parents, by dividing by k instead of 2 in equation (1)

in the random walk. Thus, a walker located at a site v has a probability $p_{v,u}$, given by (2), to move toward a child u. An algorithm that computes all the real weights consists in iterating over all the sites, from the tips to the genesis, applying the equation (1). We will denote this TSA *MCMC-new* in the remaining of the paper.

3 Analysis

Complexity The TSA is executed each time a transaction is issued by an agent, thus it should be efficient.

During its execution, the MCMC($\alpha = 0.001$) algorithm must first compute the cumulative weight of each site and then perform two random walks from the genesis to a tip. The complexity of a random walk is in $O(n)$ where n is the amount of sites in the Tangle, assuming the average number of child per site is constant. Computing the cumulative weight, however, is far from efficient, as existing algorithms are in $\Theta(n^2)$ [Gal18] with a space complexity also in $\Theta(n^2)$.

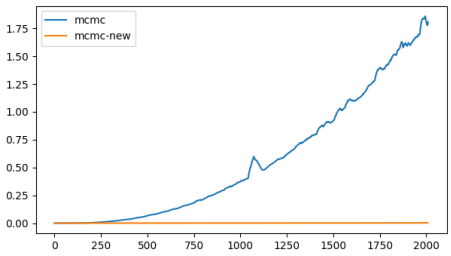

Fig. 2: Execution time in second of the TSA in function of size of the Tangle

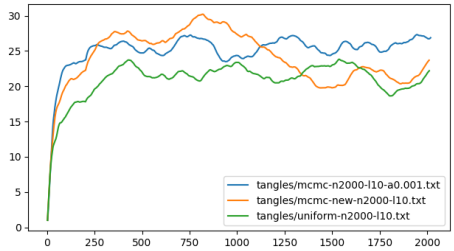

Fig. 3: Number of tips in function of the size of the Tangle

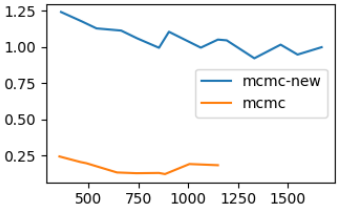

Fig. 4: Security factor in function of the number of sites confirming the target.

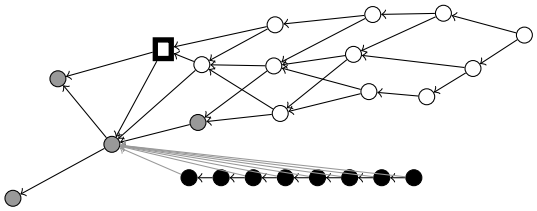

Fig. 5: Generation of a *feather* parasite Tangle. The rectangle is the target site, white sites are honest and black sites are parasite ones.

Computing the real cumulative weight only requires $O(n)$ operations. Indeed, a depth-first iteration of the Tangle is sufficient. For each site u we compute $R(u)$ in function of the value $R(c)$ of its children: $R(u) = 1 + \sum_{c \in C(u)} R(c)/2$. Figure 2 shows the evolution of the executing time of the MCMC($\alpha = 0.001$) and MCMC-new in function of the size of the Tangle.

Stability of the Number of Tips We define the stability of a TSA as a property to maintain a constant amount of tips in average. Formally, with a perfectly stable TSA, the average number of tips is a Markov chain with a non-null stationary distribution [Bra18]. We know [Pop16] that the uniform TSA is stable and the average number of tips is 2λ in average with λ being the average numbers of sites generated per time unit. By simulations, we can observe that MCMC($\alpha = 0.001$) and MCMC-new are also stable with an average amount of tips around 2λ, similarly to the uniform TSA.

Security In order to analyse the security of a TSA, we measure the amount of sites required to create a parasite Tangle. For a certain Tangle T and a target site t, a *parasite Tangle* is a set of sites P connected to T such as *(i)* each site of P does not confirm the target site t and *(ii)* the probability for the TSA to chose a tip in P is greater than $1/2$. The security factor is then the ratio between the number of sites confirming the target site and the number of sites in the parasite Tangle. Intuitively, the parasite tangle represents the sites an attacker has to generate to perform a double spend on the target, and the security factor is the proportion of the computational power an attack should own to do so.

Figure 4 shows the security factor of $MCMC(\alpha = 0.001)$ and MCMC-new in function of the number of sites confirming the target site. These results are obtained by generating Tangles of variable sizes. In order to generate our parasite Tangle, we use a feather attack (shown in Figure 5), which consists in generating sites such as each one confirms the previous one and a site confirmed by the target site. In the MCMC($\alpha = 0.001$) case, the attack is slightly different as each parasite site confirms the previous parasite site and a site among a set of sites confirmed by the target site (and not only a single site)

We observe that the security factor of MCMC-new is near 1 and is much lower for the MCMC($\alpha = 0.001$). However, we conjecture that the security factor of MCMC($\alpha = 0.001$) tends to 1 as the number n of sites that confirm the target tends to infinity. Indeed, for really big values of n, the random walk depends almost only on the subtangle's weight. One can show that in the best cast (for the adversary), the parasite Tangle is a feather and gets attached to the Tangle on a site s which has only one child in the Tangle t. When the parasite Tangle is sufficiently big, every random walk must step by s and the probability for a random walk in s to chose to go to the parasite Tangle is greater then $1/2$ if:

$$\exp(-\alpha(w_s - w_t)) < \sum_{i=1}^{k} \exp(-\alpha(w_s + i))$$

Where k is the size of the parasite Tangle and w_s, resp. w_t, is the site cumulative weight of s, resp. of the target site. The sum on the right part of the inequation is the contribution from each site of the parasite tangle, because in a feather parasite Tangle, each site is a child of s. We are interested by finding the smallest k verifying this inequation, *i.e.*such as n/k is the security factor. By calculus, we find that for n tending to infinity, this n/k tends to 1.

We showed that our algorithm performs better than the one currently used in production in the IOTA cryptocurrency, considering computational complexity and security and is equivalent in terms of stability.

References

[Bra18] Quentin Bramas. The Stability and the Security of the Tangle. working paper or preprint, April 2018. URL: https://hal.archives-ouvertes.fr/hal-01716111.

[Gal18] Alon Gal. Algorithm for calculating cumulative weight, 2018. URL: https://github.com/alongalky/iota-docs/blob/master/cumulative.md.

[Pop16] S Popov. The tangle. white paper, 2016. URL: https://iota.org/IOTA_Whitepaper.pdf.