École Normale Supérieure de Rennes

Internship Report - ICube Laboratory - Team Réseaux

Tangle analysis for IOTA cryptocurrency

Author:
Vidal Attias

Supervisor:
Dr. Quentin Bramas

August 23, 2018

Abstract

In this paper, we analyse the Tangle structure, we will present a multi-agents model, study some properties and define two algorithms to compress in an optimal way the Tangle.

Intership done from May 22 to July 20 2018 in the iCube Laboratory under the supervision of Quentin Bramas, iCube-CNRS.

Contents

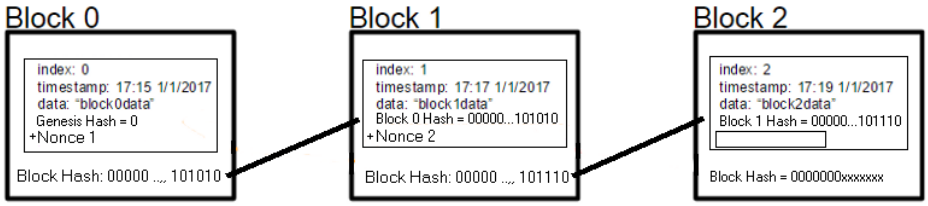


Figure 1: Blockchain schematic

1 Introduction

Distributed database is a thriving emerging new technology, which has undoubtedly a great potential to bring opportunities such as cryptocurrencies or the Internet of Things (ioT) applications. Both of them are also blooming fields which gain in importance. However, having a distributed and decentralized database has many inherent problematics such as Byzantine Fault Tolerance (BFT) [5] [1], complexity, difficulties to maintain the data integrity [3]. Nonetheless, the benefits outnumber the flaws, it allows a full transparency, availability, robustness and scalability.

In the last few years, some technologies have emerged such as the blockchain [6], a ledger composed of blocks each referring to the latest one, which aim is to provide a efficient structure for decentralized databases. The key idea is to agree on a consensus, which means being able to rely on the agents issuing data in the network. The main idea proposed is the proof of work which consists in proving that an agent has enough computation power to be considered trustworthy, i.e. spending such a power that a malicious attack is vain. However this solution has a lot a flaws, in particular the bitcoin cryptocurrency, based on the blockchain, has a dramatically expensive energy consumption [2]. Thus, there is a clear need of a new type of consensus, which guarantees security but has a low environmental footprint.

A new technology has latterly gained interest in the distributed ledger field, which is the Tangle, described in a white-paper. [7] The Tangle structure is based on a directed acyclic graph representing the ledger and the consensus relies on the confirmation rate for each site. Each node represents a transaction and an edge represents a confirmation. Then the more a transaction is confirmed by the recent ones, the more it is considered trustworthy. Nevertheless, it is a really young technology, and is not yet well understood. There is a lot of study to lead on the Tangle properties and this structure may have a really huge size with time, preventing from performing heavy computations.

This paper will be structured as following, first we will define the context and explain our model and the different parameters available, then we shall explain the results of the simulations obtained and finally we will present the Tangle compression.

2 Context

The first paper published about the Tangle, the white-paper [7], presents a directed acyclic graph in which each node is called a site. The structure works the following: each site corresponds to a transaction issued in the network, and each transaction has to "confirm" k older transactions and an edge between two sites means that the later confirms the older. The very first site present in the Tangle is called the *genesis* and may be considered as the basis of the Tangle. The subtility is that a new transaction may confirm only sites which has not been confirmed yet, we call them *tips*. The white-paper suggests to take $k = 2$ and is followed by almost the whole

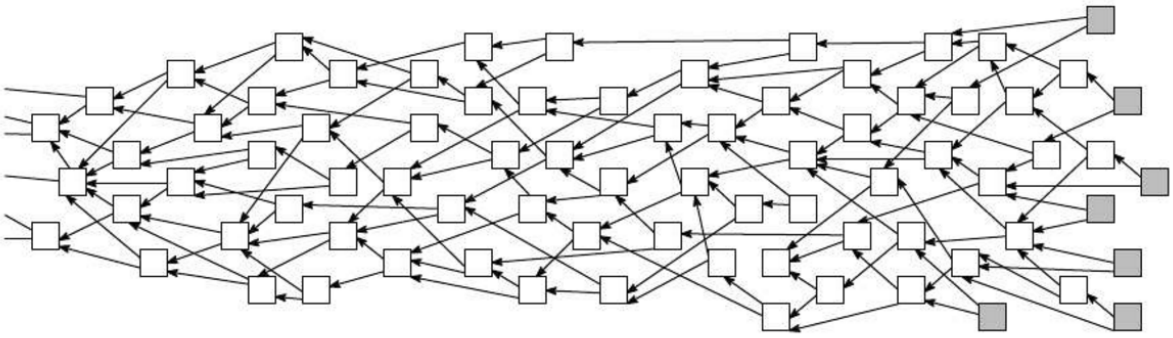


Figure 2: Tangle schematic - the tips are in grey

recent litteracy and the IOTA cryptocurrency. We say that a site indirectly confirms another one if there is a path between them in the graph.

In order to get a real graph instead of a linear graph as for the blockchain, we define an inherent latency, noted h, which represents the time of Proof of work needed to issue a transaction. A node cannot be seen as added in the Tangle during the h period, thus the tip it confirms is still considered as a tip during this period which allows another site to confirm it. The white-paper model suggests to define a constant latency for all the agents in the network, and in particular, the agent who issued the transaction is also affected by this latency.

Metrics The white-paper also defines two metrics for the Tangle, based on the weight of the nodes. We define the **self-weight** of a node as being a inherent value, often considered as dependant of the proof of work needed to issue it, but sometimes considered as constant. We then define the **score**, which is the sum of the self-weight of all the sites indirectly confirmed by the considered site and the cumulative weight as the sum of the self-weight of all the sites indirectly confirming the considered site.

Tip selection algorithm The major stake of the Tangle structure is the tip selection way while issuing a new transaction. Indeed, we do not want to impose a selection algorithm in the network, but we do want a consensus which guarantees the Tangle's security against conflict transactions which would lead to serious branching of the strucure. If an attacker has the ability to take control of the Tangle by chosing special tips, it's a critical security issue. Thus, we have to make up tips selection algorithms which, if widely adopted by the network agents, implies people not using it to be inefficient in the Tangle, which means their transactions have low chances to be widely confirmed.

Uniform algorithm The most naive algorithm is to choose uniformly the tips in the tips set. This is really cheap to compute, because an uniform distribution is really easy to generate, however it can lead to security threats, as it doesn't rely on any confidence metric, such as the score or the cumulative weight, then in case of two conflictual branches, no one would be favorised, leading an attacker to take benefit of that.

Markov Chain Monte Carlo algorithm A second algorithm, called MCMC algorithm, standing for *Markov Chain Monte Carlo*, consists of a random walker travelling the graph from

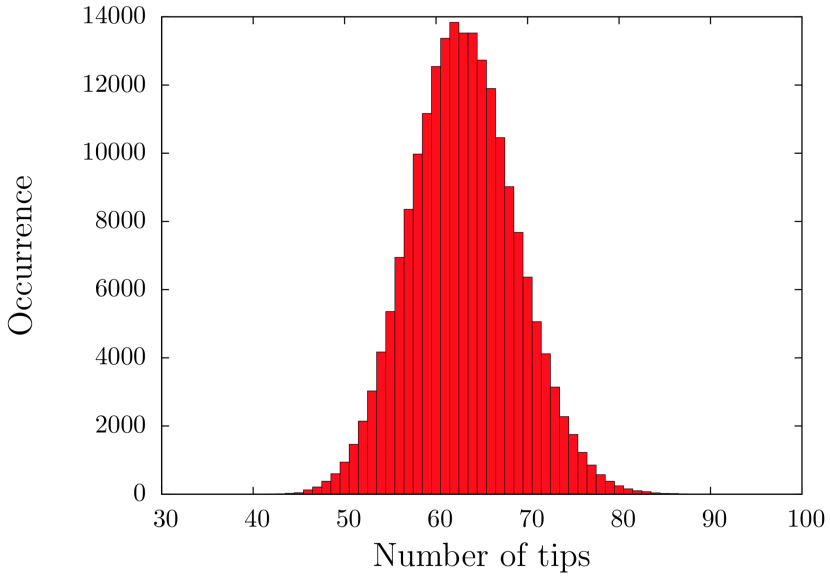


Figure 3: Stabilisation around L_0

the genesis to one of the tips. For each iteration, it chooses the next site with a probability according to its cumulative weight. The greatest the cumulative weight, the greatest the probability to jump to. This algorithm has the advantage to promote the main branch, as its cumulative weight is going to grow, however it is very sensitive to the inner topology of the Tangle, which is really complicated to predict.

Computing issues The main issues in the current context are numerous. First of all, we don't have yet a multi-agent consideration of the network, the white-paper considering a single agent issuing transactions. Secondly, the MCMC algorithms is really complicated to implement as for now, we do not have a computationally easy way to compute the cumulative weight for the sites as it changes with time. Furthermore, performing a random walk in the tangle is really expensive in termes of computation as the Tangle can easily have millions of sites in a long time.

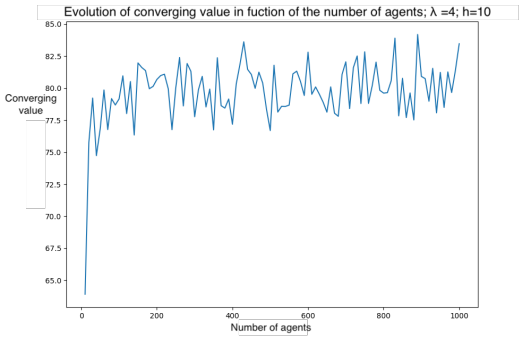
Theoretical analysis The white-paper presents then some theoretical properties wished. In particular, one should expect the amount of tips over the time to fluctuate around a convergence value. Let's take the white-paper's notations, let $L(t)$ be the number of tips over the time, the desired property is that $\lim_{t\to\infty} \mathbb{P}(L(t) = L_0)$ exists and is positive with L_0 the convergence value. The reason is to limit the "left behind" tips, i.e. the tips which are never going to be confirmed because of the tip selection algorithm's design and then breaking the Tangle's efficiency.

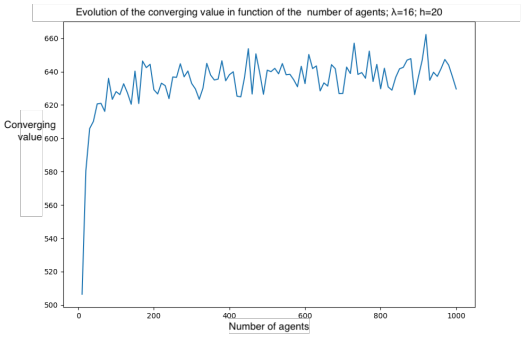
3 Multi-agents model

3.1 Contribution

As described earlier, the current litteracy does not mention any multi-agents model for the networks, as it is a major charateristic of a distributed database. We had to design a multi-models model which represents a realistic use of the Tangle.

We assume that the process of incoming transactions can be modeled by a Poisson point process. Let λ be the rate of that Poisson process. Each time a transaction has to be issued,





the simulator choses uniformly an agent and it has to chose two tips, issue a new transaction and notify all the other agents it has issued the transaction. The notification is modelized by a requests with a recipient and a delivery time, stored in a buffer, which contains the non-delivered requests. The delivery time is the current time plus the latency between the two agents. Each agent is assumed to have a certain self-latency to the network core, then the latency between two agents is the sum of their self-latencies, except that the latency between an agent and itself is zero, although we could have a design where an agent could have a self latency.

Each agent has a local view of the Tangle, which is updated for each notification received. Thus, when an agent issues a transaction, it can immediately see it, it doesn't have to wait, which allows it to confirm it immediately if it is willing to issue multiple transactions, preventing the tips size from growing too much.

We developped a C++ simulator firstly based on this design, which has been used for all the results displayed in this paper, however, this design has some flaws. The first one is computational. Indeed, have to deliver the request for each nodes is quite senseless, given that the information is strictly the same, it then slowered the execution. The second flaw is spatial consumption. Indeed, storing as many local views of the Tangle as the amount of agents is quite ponderous, while the information was heavily redundant as the first sites should be visible by the whole network after a certain time. We ended by redisigning the whole simulator to comply with the formalization defined in [4].

3.2 Validation

The validation here is based on two main parameters. First of all, we want to check if we can find out a similar behaviour than the white-paper predictions, mainly about tips amount stabilization and then study the differences that our model implies. We had only time to test the random algorith, although it would have been really interesting to test also the MCMC.

As stated by the white-paper, there should be a fluctuation of the amount of tips around a certain value. According to the white-paper, this value should be $L_0 = 2h\lambda$ in a continous model, then depending only on the inner latency and the Poisson rate of incoming transactions. When we first tested the simulator, with 10 agents, we effectively found a stabilization but with a factor significantly less than two, no matter the values of h and λ. We then realized that changing the amount of agents changes this factor. Then the amount of tips $L_0 = f(N_a)h\lambda$ where N_a is the amount of agents and f a function of N_a which remains to determine. We then decided to plot the variation of $L0$ according to Na with constant values of h and λ and the results are in 3.2.

The results of this study is that $f(N_a)$ converges to $2h\lambda$ when N_a becomes high, and reaches the convergence value approximately with 200 agents. That means that more agents should not change L_0 which is a really good property, because 200 agents is quite a low value for a

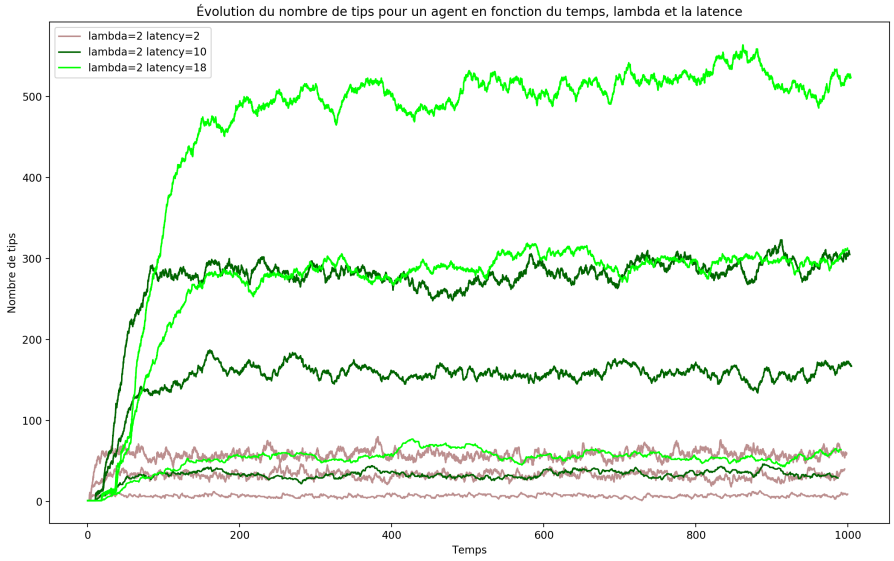


Figure 4: Evolution of the number of tips over time, λ and h

network, besides, the amount of agents may fluctuate a lot with time, this stability is then quite interesting. However, in our model, we do consider that λ is independant of N_a which is not necessarily true, but one could consider the network to settle the λ value.

We then studied the convergence time to L_0. The protocol is the following. The convergence time is defined as the time to reach 95% of L_0. Then we derive a time constant τ as in an exponential law which is a third of the convergence time. It turns out that we found a linear relation between the convergence time according h but no variation according to λ, with a factor of 5.8 thanks to 4. Then a first approximation of $L(t)$ would be $L(t) = f(N_a)h\lambda(1 - e^{5.8g(N_a)ht})$ where $g(N_a)$ would be the impact of N_a on the convergence time. There are many chances $g(N_a) = 1$ but as we haven't tested on N_a it is uncertain.

Finally, we wanted to study the differences between the local views of each agent. The main parameter we focused on was the amount of tips visible by each agent and the amount of tips actually present in the global Tangle, which can be seen as an omnicious agent aware of everything instantly. A simulation sample can be seen in 5. One can see a really distinct curve, the brown one, which represents the amount of tips in the global Tangle, the other one being the agent's tips. We can see there is really few variation between the agents, each of them have merely the same amount of tips. One remarkable thing is that there seems to be a dependance on λ. Indeed, we have set a latency of 4 seconds, and as we stripped the plot with 4 seconds width strips, we see, especially at the beginning a different behaviour for each strip. In the first one, it is hard to see, but each agent has the same amount of tips, an only one, because of the latency it can only confirm its own sites, and the global Tangle tips grows until reaching the value n, with n being the number of agents, which is logical, the global Tangle beeing the genesis and n linear branches because of the latency. One should notice that the global Tangle's tips is reaching n because we have h great enough to have an agent to generate a site and thus adding its own branch to the Tangle. Afterward, the amount of tips can increases greatly because the agents can confirm already confirmed tips because of the latency, then adding a tip to the global Tangle without removing a former one and finally stabilizes around a converging value, as predicted in [7].

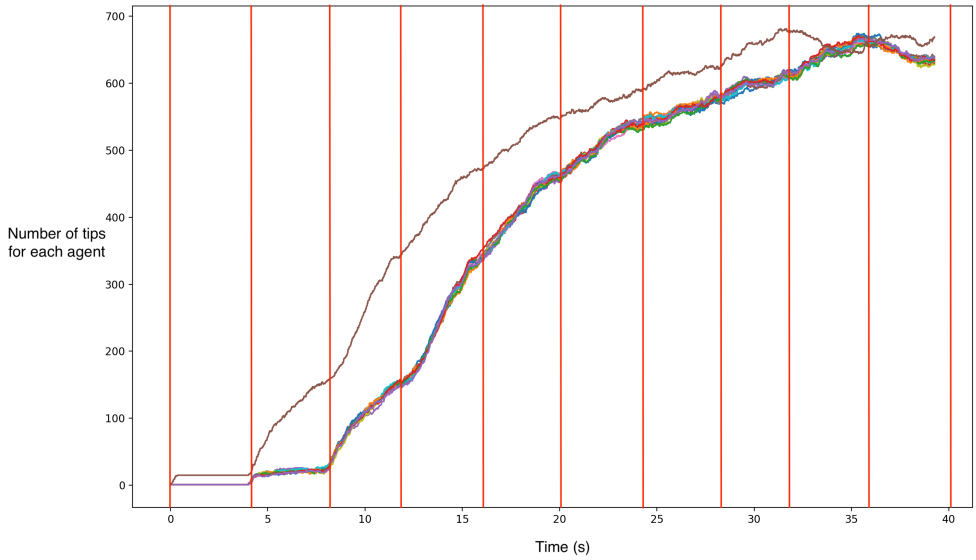


Figure 5: Amount of tips for each agent over the time

There is one really interesting thing, the global Tangle's line seems to predict the behaviour of the agents for the next strip, which is unexplained yet, but could be a really interesting research.

4 Compression algorithm

4.1 Contribution

As explained in the introduction, we have critical issues in implementing the MCMC tips selection algorithm as computing the cumulative weight and performing a random walk are really expensive operations. Many ideas have come up, as computing a look-alike cumulative weight or performing the random walk form the middle of the Tangle. However, these solutions also present flaws in terms of security and accuracy.

We are presenting in this section a way of compressing the Tangle into a really short new Tangle, having the same tips set and conserving the most important properties of our tangle. Firstly, the compressed Tangle preserves the original accountancy, which means that for each tip, the confirmed transactions are strictly the same, and secondly, it preserves the global topology of the original Tangle and allows really similar random walks.

The main idea of the compression, is to agregate the existing sites according to the set of tips confirming them, i.e. creating equivalence classes. As the major part of the sites should be gathered in really long branched, a whole branch could be resumed as a single site.

We suggest here two algorithms, which have a different design but have almost the same compressed Tangle, with a subtil difference. The construction of the new sites is the same, their are in both cases the equivalence classes of confirming set of tips. The difference lies in how they connect the edges. On the one hand, we have an algorithm which makes an edge between two new sites if there is an edge between a site of each of them in the original Tangle, on the other hand we have an algorithm which takes benefit of a particular property of inclusion of the new sites.

Formalization We define the notation $A \rightsquigarrow B$ which means "A indirectly confirms B". In other words, there is a path between A and B.
Let's consider our Tangle as a directed graph $G = (S, E)$, S being the sites set and $E \subseteq S \times S$ the set of edges. We besides define $T \subseteq S$ the tips set, i.e. $T = \{t \in S \mid \forall s \in S, (t, s) \notin E\}$.
Let's define the application $C : S \to \mathcal{P}(T)$ such as $C(s) = \{t \mid t \in T, t \rightsquigarrow s\}$, i.e. associates to each site the set of tips confirming it.
Now we build $\mathcal{C} = \{C(s) \mid s \in S\} \subseteq \mathcal{P}(T)$ the set of equivalence classes in the Tangle.
Now we define the "inverse function" of C, which is $C^{-1} : \mathcal{C} \to \mathcal{P}(S)$, such as $C^{-1}(c) = \{s \mid C(s) = c\}$.
Thus, we call our new compressed Tangle's $G' = (S', E')$ and we have $S' = \mathcal{C}$. The remaining set, E' will be generated by either of our algorithms.

By existing edges As described earlier, in this algorithm, we are going to connect the former connected equivalence classes. More formally, $(c, c') \in E'$ iff $\exists\ s \in C^{-1}(c), s' \in C^{-1}(c')$ such as $(s, s') \in E$.

By inclusion In this algorithm, we use a property of the compressed graph which is, $c_1 \rightsquigarrow c_2$ iff $c_1 \subset c_2$ where $c_1 \neq c_2$, which will be demonstrated later.
We begin by sorting $\mathcal{C}$ in a descending order, and then we add

Algorithm 1 Site.Append()

Require: S the set to add
  Bool appendedToChild ← False
  for all child in childs **do**
    if $S \subseteq$ child **then**
      appendedToChild ← True
      child.Append(S)
    end if
  end for
  if $S \subset$this **and** appendedToChild == False **and** S not in childs **then**
    childs.append(S)
  end if

Algorithm 2 Compressed Tangle construction

Require: G' the equivalence class of the original genesis, S the set of all equivalence classes in an descending order in terms of cardinality
  for all s in S **do**
    G'.Append(s)
  end for
  return G'

Cumulative weight Computing the cumulative weight with the compressed Tangle is really efficient. First, we have to define the self-weight of the new sites. The weight of each of the new sites is the sum of the self-weight of each original sites now composing the new one. More formally, $\forall s' \in S', H(s') = \sum_{s \in C^{-1}(s')} H(s)$. One may understand here quite easily that computing the cumulative weight is way more faster, since computing the self-weight of the new sites takes a constant time while compressing, and as the cumulated weight algorithm on the compressed

Tangle is the same as the original ones, then if the compressed Tangle is small enough, then performing it would be far less expensive, especially if the algorithm is not linear, say quadratic.

4.2 Validation

Proof of correctness We need to have formal proofs of our algorithm in order to have guarantees that our compression does work efficiently.

Definition 1. *We define the operator* $\bar{.}$ *as following :*

$$\forall t \in T', \bar{t} \text{ is the only element in } t$$

In other words, this operator does give for a certain tip present in the compressed Tangle the only original tip.

Let us begin to define a new property, which is the accountancy correctness. We say a compressed tangle $G' = (S', E', T')$ is and accountant compressed Tangle of the Tangle $G = (S, E, T)$ and we note $G' \models G$ if each tip validates exactly the same transactions.

Definition 2. *We say* $G' \models G$ *if* $\forall t \in T', \forall s' \in S', (t \rightsquigarrow s' \iff \forall s \in C^{-1}(s'), \bar{t} \rightsquigarrow s)$ *which means that every site validated by a site has to be validated by it in the compressed Tangle.*

Lemma 1. $\forall s, s' \in S, s \rightsquigarrow s' \implies C(s) \subseteq C(s')$

Proof.
Let $s, s' \in S$. We have defined $C(s) = t \in T \mid, t \rightsquigarrow s$.
Let $t \in C(s)$, then $t \rightsquigarrow s$, with $s \rightsquigarrow s'$, then $t \rightsquigarrow s'$.
We obtain that $t \in C(s')$
Hence $C(s) \subseteq C(s')$

■

4.2.1 By existing edges

The Tangle's main property to preserve is the accountancy

First way

Proof.

Let $t \in T'$, $s' \in S'$ such as $t \rightsquigarrow s'$.
Let us show that $\forall s \in C^{-1}(s'), \bar{t} \rightsquigarrow s$**.**
We have $t \rightsquigarrow s$, then $\exists n \geq 2, \exists c_1, \ldots, c_n$ with $c_1 = t, c_n = s'$ such as

$$c_1 \longrightarrow c_2, \ldots, c_n \longrightarrow c_n$$

Let us now show by *induction*, $\forall k \in [\![2, n]\!], \forall c \in C^{-1}(c_k), \bar{c_1} \rightsquigarrow c$.
Base case: $k = 2$
We have $c_1 \longrightarrow c_2$, then $\exists s_1 \in C^{-1}(s_1), s_2 \in C^{-1}(s_2), s_1 \longrightarrow s_2$ by definition.
Then $C(s_1) \subseteq C(s_2)$ then $\bar{c_1} \in C(s_2)$.
Then $\forall c \in c_2, \bar{c_1} \longrightarrow c$.
Step case: $k \in [\![2, n-1]\!]$ Let us assume $\forall c \in c_k, \bar{c_1} \rightsquigarrow c$, we have to show that $\forall c \in c_{k+1}, \bar{c_1} \rightsquigarrow c$.
We have $c_k \longrightarrow c_{k+1}$, then $\exists s_k \in C^{-1}(s_k), s_{k+1} \in C^{-1}(s_k), s_k \longrightarrow s_{k+1}$ by definition.
Then $C(s_k) \subseteq C(s_{k+1})$ then $\bar{c_1} \in C(s_{k+1})$.
Then $\forall c \in c_{k+1}, \bar{c_1} \longrightarrow c$.
Conclusion We can conclude here, $\forall s \in C^{-1}(s'), \bar{t} \rightsquigarrow s$.

■

Second way

Proof.

Let $t \in T$ and $s \in S$ such as $t \rightsquigarrow s$.
Let us show that $C(t) \rightsquigarrow C(s)$.
We have $t \rightsquigarrow s$, then $\exists s_1, \ldots, s_n$ such as

$$s_1 \longrightarrow s_2, \ldots, s_{n-1} \longrightarrow s_n$$

Then $\exists c_1, \ldots, c_r \in S'$, with $r \leq n$, $c_1 = C(t)$ and $c_r = C(s)$ such as

$$c_1 \longrightarrow c_2, \ldots, c_{n-1} \longrightarrow c_n$$

Conclusion: we have $C(t) \rightsquigarrow C(s)$.

■

4.2.2 By inclusion

Unfortunately, we haven't been able to perform the proof of this algorithm because we changed lately the formalization, thus making the proofs all over again was time consuming. However, the proof can be led using a loop invariant, each time we add an edge, we can prove that for the resulting temporary Tangle G, there is a subtangle $\tilde{G}$ of the original one such as $G \models \tilde{G}$ and prove that at the end of the loop, the subtangle is the whole original one. However, this proof is for a later research.

Simulations We have performed some simulations in order to test the compression efficiency. We have focused on the compressed Tangle's size, thus there are no differencies between the two algorithms.

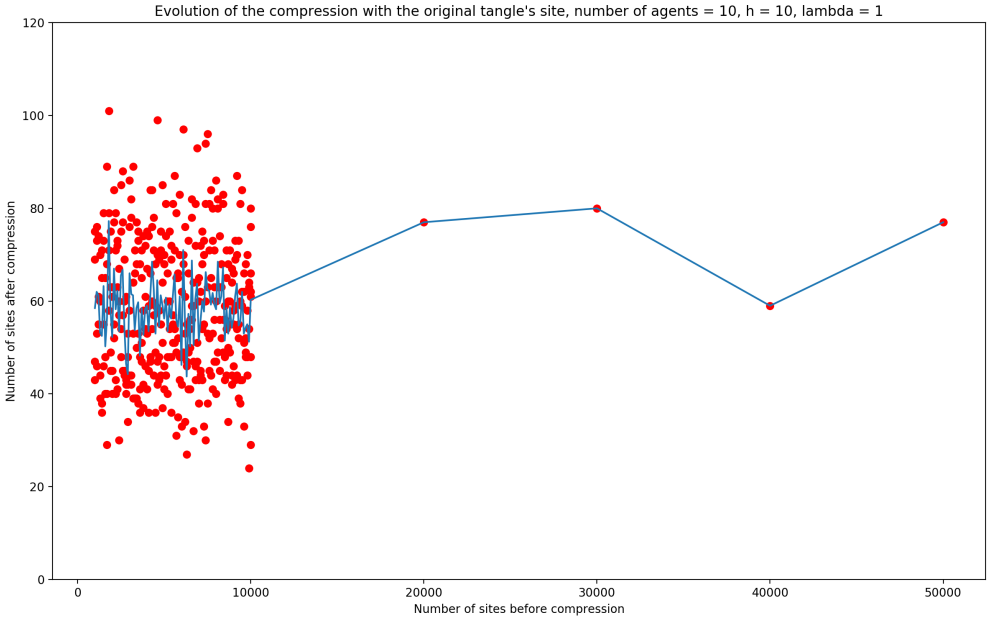


Figure 6: Compressed Tangle's size in function of the original size

We have tried to test the compression's limits particuliarly the evolution of the compressed tangle's size with the original tangle's size. In the figure 6, one can see there seems to be a stabilisation, which is quite intuitif, because the equivalence classes may join at a certain time.

In the figure 7, we have tried to see the evolution according to λ and h. However, although we can see an evolution for low values of λ and h, we can see a fall then a plane of really low values. This is due to our simulation time which was too small, yielding to tangles not grown enough to have an efficient compression. The we shall run other simulations, but they are really heavy and the current plot remains interesting.

Algorithm's limits Although being performant for specified criteria in the Introduction, this algorithm has some issues. Firstly, the computing complexity for the connecting edges version is quite heavy, because of a multiple graph iteration.

Furthermore, there is a lot of information loss. Indeed, one does not know anymore which site contained which transaction inside an equivalence class, everything is all mixed up. We could save these informations by using a data structure saving them in the new sites, but there would be memory growth.

Besides, this algorithm is statical, it doesn't compress the Tangle on the fly, and also we have to keep in mind that the set of tips a agent sees at a certain time is not necessarly the same the others see because of the latency, thus, performing this algorithm could lead to serious problems. We have some solutions here.

We could perform the algorithm at a certain time and compress the Tangle at a state it was at the current time minus a "safety time". The safety time being the delay to be sure each change in the graph seen by an agent has been also seen by everybody. In our model, this is quite simple to do, because the latency is constant, however, it could be dangerous for real life situations.

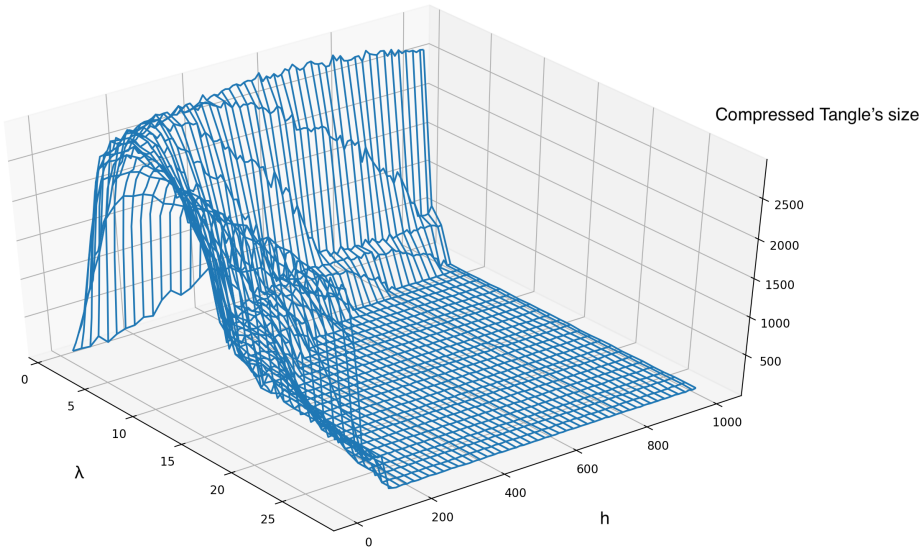


Figure 7: Evolution of the compressed size in function of λ and h

The second solution would be to keep a trace of each existing site and store it in the compressed sites in order to know to which new site connect an edge when a new transaction arrives.

Another problem is to know when and how to perform the algorithm. The point is that having a compressed Tangle is only an accomodation for the agents and it is still compatible with the non-compressed Tangle if the right structure is chosen. This means that if an agent has decided to compress the Tangle and the rest of them not, it can still work with them and exchange transactions. However, we could define that some agents in the network are dedicated to compressing the Tangle and broadcasting it in the network, but there would be consensus and security issues. Thus, we think that the compression should be the very own choice of the angents. Regarding the compressing frequency, we do not have answers yet. As the compressed Tangle's size seems to be constant, this operation should be constant in terms of computation complexity, then performing the algorithm really frequently might not be a bad idea. A last idea would be to consider the compression as a *proof of work*.

5 Conclusion

In this paper, we have presented a multi-agents network model, allowing us to show that although there is a variation with the number of agents, there could be stabilization around 200 agents, which means that for a real implementation, there should not be scalability issues. Furthermore, the convergence time, i.e. the time to reach Tangle stabilization seems to be independant of the temporal rate of transactions, which is the most complicated parameter to enforce in our model. This means that a change of parameters in the network, such as h or λ, should not induct a variation for a time longer than the stabilization time.

We have also presented two compression algorithms, which compess the Tangle in order to perform faster cumulative weight calculus and MCMC tip selection algorithm. The most

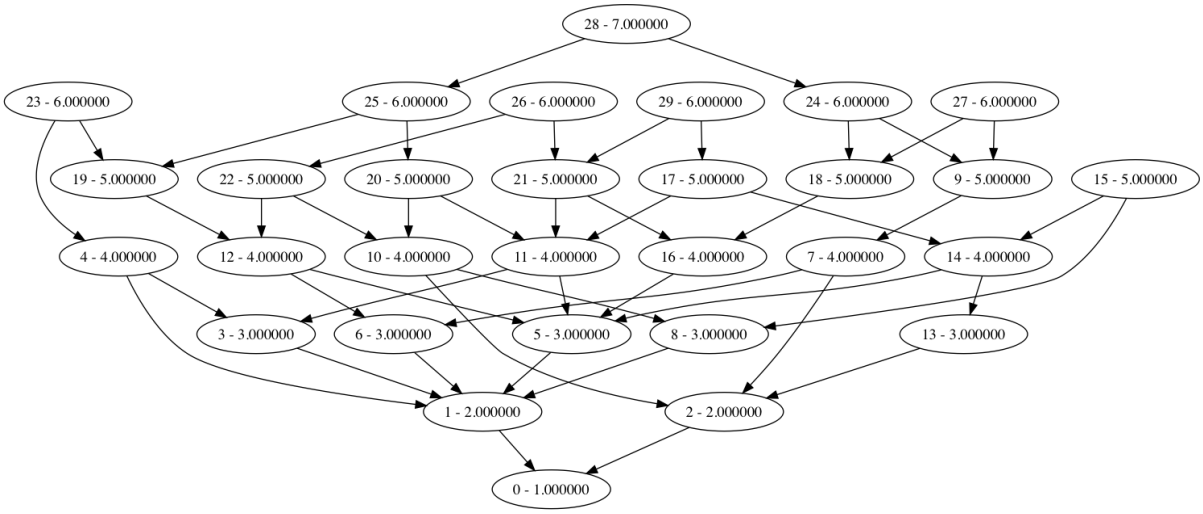


Figure 8: Original non-compressed Tangle

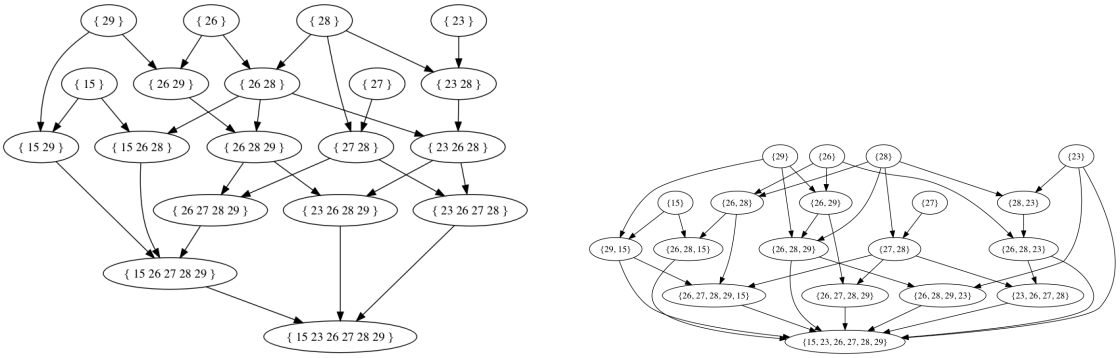


Figure 9: Comparison between the inclusion and the existing edges algorithms

interesting property empirically found is that the size of the compressed Tangle seems to fluctuate around a certain value, which is really attractive, meaning that the MCMC algorithm can be considered as performed in a constant time.

Many gray areas are remaining. Concerning our multi-agents model, there is a lack of simulations according to the parameters, especially concerning the convergence time according to the number of agents, and the stabilization of the amount of tips in function of the number of agents. Furthermore, a study of the other tip selection algorithms is mandatory, as it could reveal other behaviors.

Concerning the compressing algorithm, we have to perform too many simulations, varying the parameters to understand their impact. This is just a blueprint of the algorithms. We would also like to test on the real IOTA data, in order to check the viability of the algorithm.

Acknowledgements

I would like to thank the ICube laboratory for welcoming me and letting me a great scientific independance and always encouraging me in my research, and especially Quentin Bramas and Thomas Noel for being really present and helpful.

I would also like to acknowledge the IOTA's foundation help, especially Alon Gal for sharing with me unpublished papers dealing with my multi-agent model, which have helped me to rectify some details, and for reading my drafts about the compression algorithm and taking time to discuss about it.

Finally, I want to thank the École Normale Supérieure for giving me the opportunities to develop a scientific state of mind which has allowed me to get into this internship with serenity.

References

[1] Alysson Bessani, João Sousa, and Marko Vukolić. A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In *Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, SERIAL '17, pages 6:1–6:2, New York, NY, USA, 2017. ACM.

[2] Alex de Vries. Bitcoin's growing energy problem. *Joule*, 2018.

[3] Dr.S.B.Kisho D.S.Hiremath. Distributed database problem areas and approaches. *IOSR Journal of Computer Engineering (IOSR-JCE)*, 2016.

[4] Alon Gal and Clara Shikhelman. Partitioning in the tangle: a multi-agent extension. submitted.

[5] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[6] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf.

[7] Serguei Popov. The tangle. `https://www.iotatoken.com/IOTA_Whitepaper.pdf`, 2016.