Multi-agent Based Simulations of Block-free Distributed Ledgers

Michele Bottone, Franco Raimondi, Giuseppe Primiero
Department of Computer Science
Middlesex University, London
e-mail: {m.bottone,f.raimondi,g.primiero}@mdx.ac.uk

***Abstract*—In the past ten years distributed ledgers such as Bitcoin and smart contracts that can run code autonomously have seen an exponential growth both in terms of research interest and in terms of industrial and financial applications. These find a natural application in the area of Sensor Networks and Cyber-Physical Systems. However, the incentive architecture of blockchains requires massive computational resources for mining, delays in the confirmation of transactions and, more importantly, continuously growing transaction fees, which are ill-suited to systems in which services may be provided by resource-limited devices and confirmation times and transaction costs should be kept minimal, ideally absent. We focus on a new block-less, fee-less paradigm for distributed ledgers suitable for the WSN, IoT and CPS in which transactions are nodes of a directed acyclic graph, that overcomes the limitations of blockchains for these applications, and where e.g. sensors can be at the same time issuers of transactions and validators of previous transactions. In particular, we present and release open-source a simulation environment that can be easily extended and analysed, and confirms the available results on the performance of the network.**

***Keywords*-transactions; blockchain; DAG; tangle; simulations**

I. Introduction

The rise of billions of connected and relatively autonomous systems in the past few years provides a useful backdrop for distributed computation. Typically, these fall under the umbrella terms of Wireless Sensor Networks (WSN), Internet of Things (IoT), and Cyber-Physical Systems (CPS). Three characteristics of these autonomous systems stand out: they consist of many components, such as sensors, actuators, controllers; they exchange information, often under resource constraints; and they are dynamic in time or space. In these systems, the interplay between information, decisions, actions, and trust is a worthwhile subject of study under a multi-agent perspective [1], [2]. A ubiquitous example is *credit money* [3], which transfers trust between economic agents and whose usefulness arises as a network externality that facilitates transactions [4], [5]. Cryptocurrencies attempt to replicate this memory function and other characteristics of real-world money such as safety and consistency. Early attempts at cryptographic cash relied on trusted authorities that maintained centralised ledgers, such as banks and credit card companies. The main technological advance of Bitcoin [6] was to introduce a distributed ledger secured by the majority rule without any central authority, by means of a so-called *blockchain* and standard cryptographic primitives like signatures and hash functions. It also introduced the possibility of transaction scripting as a way of enabling smart contracts and micro-transactions based on distributed architectures which are suitable for CPS and the IoT, and spurred academic and practical interest.

The blockchain used by Bitcoin and its descendants is a time-linear data structure: transaction data is stored in *blocks* which must each contain a reference to the block that came before it. Blocks are created by specialised users called *miners* that perform a cryptographic proof-of-work (PoW). It is natural to consider the temporal succession of blocks as a directed flow of transactions linked together by consensus. The blockchain algorithm has proved resilient to attacks and double spending, suffering only setbacks in coin exchanges. However, as a cryptocurrency for lightweight systems in the IoT, Bitcoin has two serious drawbacks which make it unsuitable for micro-payments; firstly, because of the dependence on miners for processing and verification, the transaction fees are relatively high and rising, especially in low throughput; secondly, it scales poorly since the blockchain network performance degrades in the number of users. It has become apparent that the scale and fee issues experienced by Bitcoin are intimately connected to the mining system and its incentive structure, which under consolidation of miners, reduce the degree of decentralisation of the network. *Mining pools* now make up over 90 percent of the hashpower in the Bitcoin network, and tend to be heavily concentrated geographically. Verification delays for reaching consensus are the norm: because of the fixed MB limit on the blocks, Bitcoin currently processes $3/4$ transaction per second (txps) and is capped at 7 txps, while Visa and MasterCard are capable of processing 60000 txps. Security of the protocol has also been called into question when miners can collude or are exposed to geopolitical risk.

A promising avenue of recent research and development has involved blockchain-free currencies. In these approaches, the blockchain and its consensus algorithm are replaced by a directed graph of cross-verifying transactions based on the mathematical properties of a *Dyrected Acyclic Graph (DAG)*, which serves as a truly distributed ledger and, generally speaking, reaches consensus by accumulation of information about the state of the network. The essential idea is that to issue a transaction, users of the distributed ledger must work to approve other transactions, thus checking for conflicts and double spending, and when a transaction receives additional approvals by the chain of ensuing transactions, it becomes accepted by the system with a high degree of confidence. In

a DAG, each node represents a transaction and each edge a reference, or approval, of some other transaction in a specific direction. Such graphs are usually built up from an initial parent root called the *genesis* transaction and evolve according to precise rules, representing in this sense a lightweight generalisation of Blockchain (for an alternative construction where the direction of approval is reversed from parent to child transactions, see [7]). Several DAG-based cryptocurrencies have been recently independently proposed and implemented, among them RaiBlocks [8], DagCoin, Byteball [9] and Iota [10], which deviate from each other in the details of implementation and consensus protocols. RaiBlocks achieves consensus by using a deterministic block-lattice structure where each account has its own balance-weighted blockchain which resembles the account's transaction and balance history, and can only be updated by its owner, similar to SPECTRE [11] with restrictive permissions. Byteball reaches consensus by using a main chain of honest, trusted witnesses that reference one or more previous transactions via a MCMC selection procedure for referrals. Iota's consensus model is based on the cumulative PoW of stacked transaction where two previous transactions with low weight are selected. The latter implementation, based on the mathematical construct called the *Tangle* [12], is particularly simple to describe, yet flexible and robust to use, and has several attractive features that make it well suited for CPS and IoT.

In this paper we focus on the nature of distributed ledgers, and in particular the Tangle DAG, as *accumulated information flow*. In Section II we describe its mathematical structure, attachment and consensus model and update rules and strategies; in Section III we present an extensible, open-source multi-agent simulation environment for DAGs built in NetLogo and provide results in context; in Section IV we conclude.

II. Preliminaries

We briefly describe the mathematical object underlying the Tangle as in the whitepaper [12]. This can be thought of as a dynamic process on the space of oriented, rooted DAGs, which grows in time according to a Poisson clock for the flow of arrivals where new nodes (i.e., new transactions) are continually attached to the graph to locations which are chosen according to specific rules, and no nodes or edges are ever deleted.

A. The Tangle Process

Specifically, the Tangle is a graph $\mathcal{T} = (V, E)$, where as customary V is the set of nodes and E the set of edges and for each $v \in V$, the in-degrees and out-degrees are specified by $d_{\mathbf{in}}(v) = \sharp\{e = (c_1, c_2) \in E : c_2 = v\}$ and $d_{\mathbf{out}}(v) = \sharp\{e = (c_1, c_2) \in E : c_1 = v\}$, with $\sharp$ denoting set cardinality. For $v_1, v_2 \in V$, v_1 *approves* v_2 if $(v_1, v_2) \in E$, written $v_1 \leftarrow v_2$. Let $\mathcal{A}(u) = \{v : (u, v) \in E\}$ be the set of nodes approved by u. If there exists a covering chain $u = v_0, v_1, \ldots, v_k = v$ such that $v_j \in \mathcal{A}(v_{j-1})\, \forall j = 1, \ldots, k$ forms a directed path between them, we say that u *indirectly approves* v. The set where $\{v : d_{\mathbf{in}}(v) = 0\}$ is special: we call such nodes *tips*. The following additional rules apply: (a) within the set of all possible DAGs, each graph $T \in \mathcal{T}$ is finite and with out-degree edge multiplicity at most 2, i.e. $\forall v \in V, d_{\mathbf{out}}(v) \leq 2$; (b) there exists a *genesis* root $\rho \in V$ such that $d_{\mathbf{out}}(\rho) = 0$ and $d_{\mathbf{out}}(v) = 2\, \forall v \in V \backslash \{\rho\}$; (c) any other node $v \in V$ references ρ, i.e. there is an oriented path of approvals from this node to the genesis ρ; and (d) there are *no cycles*, i.e. paths of the type $v = v_1, \ldots, v_k = v$ for any v and k. From these assumptions, it necessarily follows at any time t the state of the Tangle $\mathcal{T}(t)$ can be concisely described by a sparse *adjacency matrix*, which is strictly lower triangular; thus the state of this matrix over time is given by a first row of 0s, a number of rows with 1s in the first column for each genesis transaction, and rows of two 1s to the left of the diagonal thereafter. This can be efficiently stored in *adjacency lists*, namely a collection of nodes and nonzero edge positions [13]. More generally, the Tangle is a continuous-time stochastic process on the space $\mathcal{T}_\infty = \cup_1^n \mathcal{T}_i \cup \mathcal{T}_{n+1} \cup \ldots$ with initial state given by $V_{\mathcal{T}}(0) = \rho$, $E_{\mathcal{T}}(0) = \emptyset$ and evolving according to the following rules:

- As a result of the flow of new transactions, the tangle grows in time, i.e. for any two times $0 \leq t_1 < t_2$, $V_{\mathcal{T}}(t_1) \subset V_{\mathcal{T}}(t_2)$ and $E_{\mathcal{T}}(t_1) \subset E_{\mathcal{T}}(t_2)$.
- For a fixed mean transactions per unit time $\lambda > 0$ the Poisson process $\Lambda(t) := \mathrm{Pois}(\lambda)$ gives the incoming transactions that then attach to $\mathcal{T}(t)$.
- Each transaction chooses two nodes v_1, v_2 and attaches a new node v to $\mathcal{T}$ with oriented edges $v_1 \rightarrow v$ and $v_2 \rightarrow v$, so that each tip unites to the set of nodes and edge points of $\mathcal{T}$ (two-edge-multiplicity rule).

This kind of DAG-based process can also be generalised to unary or n-ary out-degree multiplicities.

B. Consensus by Cumulative Weight

The Tangle achieves consensus by attaching to every transaction a positive integer[1], within some specified bounds and time limits. The basic idea is that a transaction with a higher weight is more important than a transaction with a lower weight in deciding attachment. With this in mind, we can define on $\mathcal{T}$ the partial order with respect to approvals: $\mathcal{P}^x_{(t)} = \{y \in \mathcal{T}(t) : y \leftarrow\!\!\!\circ\, x\}$, $\mathcal{F}^x_{(t)} = \{z \in \mathcal{T}(t) : z \leftarrow\!\!\!\circ\, x\}$ such that we successively refine (or zoom into) a past $\mathcal{P}_{(t)}$ and expand to a future $\mathcal{F}_{(t)}$ with respect to node x at a time t when they become attached. We now describe the consistency and tip selection process in more detail. Define the *cumulative weight* of node x as $\mathcal{H}^x_{(t)} = 1 + \sharp\{\mathcal{F}^x_{(t)}\}$, which increases with the number of nodes that directly or indirectly reference it. For any $t > 0$ if $y \leftarrow\!\!\!\circ\, x$ then one necessarily has $\mathcal{H}^x_{(t)} - \mathcal{H}^y_{(t)} \geq 1$, which implies that $\mathcal{H}^y_{(t)} = 1$ if y is a tip. We say that a transaction gets *confirmed* when it reaches a threshold θ which is sufficiently high when relative to the network usage and load. Ideally, we would like the graph to grow so that, eventually, all issued transactions are confirmed.

[1]Currently, Iota uses 3^n, but for simplicity one can assume that each node has weight 1 to start with.

In real-world networks such as those used in the Iota platform, each incoming node gets to decide which transaction gets orphaned or approved, thus propagating the Tangle in time; see `tangle.glumb.de` for a visualisation showing the consensus model based on live transaction data. It is evident that one can without loss of generality assume the Markov property, as each successive state of the process will depend only on the current state of the Tangle. In the Iota whitepaper [12], it is suggested that the optimal growth is obtained by evaluating some statistics about the transactions, which basically amount to updating their cumulative weights.

C. Tip Selection Strategies

Let $\mathcal{L}(t)$ be the set of all vertices that are tips in the tangle at time t, and let $L(t) = \sharp\{x \in \mathcal{T}(t) : \mathcal{H}^x_{(t)} = 1\}$ be its cardinality. Note that in general, $\mathcal{L}(t)$ can be decomposed into *visible* and *hidden* tips due to network delays.

Ideally, one would like the stochastic processes for both the Tangle and $\mathcal{L}(t)$ to be well behaved in the limit of a large number of transactions. In [12], theoretical considerations are advanced for $\mathcal{L}(t)$ to be positive recurrent as $t \to \infty$, i.e. $\mathbb{P}[L(t) = k] > 0$ for $k \geq 1$, rather than transient or escaping to infinity, which would leave many unapproved transaction orphaned. In practice, little is assumed in the implementation of the distributed ledger, apart from the strict approval rule, i.e any new transaction must reference two other transactions (tips) already in the Tangle. It is entirely possible that two transactions, each posted by different users, or by the same user in a short time frame, reference the same tips as each other; and in fact this happens all the time because of network latency[2]. A node can choose tips in any way it finds convenient. A particularly lazy node might try to approve a fixed pair of very old transactions, without penalty, thus not contributing to the tip approval process and increasing the likelihood that some of these might be orphaned. "Anything goes" effectively renders the Tangle a random graph.

Random Tip Selection. The simplest strategy is for each new node to select two tips uniformly at random from the list of available tips, and approve them. While conceptually simple, this strategy has the disadvantage that it does not sufficiently protect against lazy or malicious nodes. Under this hypothesis, in the steady state $L(t)$ should fluctuate around $L_0 = 2\lambda\Delta t$ in any interval Δt. The drawback of this strategy is that, especially for low network load, it can lead to *opportunistic* behaviour in that lazy nodes may decide to repeatedly select the same transaction to attach new ones, thus reducing the overall number of tips. This kind of structure can often be seen in Tangle visualisers. It can also lead to *malicious* behaviour, where users try to use their own algorithm to select tips to spam the network – artificially inflating the number of tips by issuing many transactions that approve a fixed pair of transactions – and take it over, or attempt a double spend of funds – growing for example a parasite chain with the usual attachment rules, and then attaching to the Tangle a "conflicting" transaction.

MCMC Selection Algorithm. A more sophisticated strategy to encourage "optimal" growth of the Tangle is to use a particle filter or Markov Chain Monte Carlo (MCMC) algorithm to select two tips on the Tangle [12], [14]. This still selects at random, but introduces a bias towards "honest tips" by means of an exponentially tilted random walk on the nodes of $\mathcal{T}(t)$, which become the sites of walkers that walk the reverse-directed links from the genesis ρ (or any other cutset with equal cumulative weight) to the tip. Note that while the out-degree of a node is fixed, the in-degree has a distribution, and each node may use its own pseudorandom number generator to simulate the walks. Essentially, the typical MCMC strategy will be:

(1) Choose a cutset, or suitable interval of nodes in chronological order. Usually the particle walk starts at ρ or somewhere else deep in the Tangle; if the walk does not start at the genesis, we set $q \geq 0$ as the backtracking parameter.
(2) Independently place N particles on that cutset.
(3) Let them perform a random walk, with a transition from x to y only possible if y approves x. (Optionally, repeat the walk, or select a high number of walkers N, if the two selected tips are not distinct).
(4) The transition probabilities $\mathbb{P}^f_{xy}$ between two nodes sharing a directed edge $x \hookleftarrow y$ are proportional to some monotone, non-increasing function f of the difference in cumulative weights $\mathcal{H}^x_{(t-h)} - \mathcal{H}^y_{(t-h)}$[3]. The particle is stopped when it hits a node $v \in \mathcal{L}(t-h)$[4]. Usually, $f(s) = \exp(-\alpha s)$ with $\alpha \geq 0$ having the meaning of inverse temperature (or measure of randomness). For any y and x one has the Boltzmann-Gibbs distributions:

$$\frac{(1-q)\exp\left[-\alpha\left(\mathcal{H}^x_{(t-h)} - \mathcal{H}^y_{(t-h)}\right)\right]}{\sum_{z:x\in\mathcal{A}(z)}\exp\left[-\alpha\left(\mathcal{H}^x_{(t-h)} - \mathcal{H}^z_{(t-h)}\right)\right]} \qquad x \in \mathcal{A}(y) \tag{1}$$

where $\mathcal{A}(\cdot)$ is as defined previously.

Intuitively, such a strategy spreads the approval process evenly along the most recent tips in the Tangle. If α approaches 0, this strategy is equivalent to the uniform random selection strategy; for high α it will tend to pseudo-deterministically assign high probabilities to fixed paths indexed by the highest cumulative weight, and the number of tips will grow linearly in each time step. A transaction is confirmed with a sufficiently high confidence level $\iota_0 \approx 1$ if in the low temperature regime, the walk ends in a tip that references the transaction. The rationale for choosing this type of selection strategy compared to a simpler random tip choice is that in a well-behaved Tangle, the hashing power of the network – measured by large increases in cumulative weight – is higher than that

[2]One tip can be selected in good faith by two different nodes at the same time until "snapshotting", i.e. persistent storage.

[3]In general, the node that issues the transaction might only know the state of the tangle network with a delay, $\mathcal{T}(t-h)$.

[4]If $h > 0$, it might not even be a tip anymore.

of an attacker that tries to attach a long parasite chain of transactions, whose cumulative weights would necessarily be much smaller than the sites they reference, and thus parasite sites would have a low transition probability from the main sites of the Tangle. See Section 4.1 of [12] and [14] for a game-theoretic justification.

Note that the Tangle $\mathcal{T}(t)$ as constructed induces a continuous time transient Markov Chain with large state space even for a fixed time t. This effectively means that the corresponding adjacency and transition matrices suffer from combinatorial explosion, and expanding access times[5].

D. Mean Tip Approval Times

By assumption, the Poisson clock leads to exponentially-distributed inter-arrival times between consecutive transactions: if we hypothesise λ to be the number of new arriving transactions per unit time, the mean arrival time will be on average $\frac{1}{\lambda}$ and will have a waiting-time of 0 as its mode. Thus[6] the "wider" $\mathcal{T}(t)$ is (the larger it scales in terms of the incoming transactions and number of users) the more instantaneous we can expect the Tangle propagation to be. Straightforward statistical analyses of the public data from [10] confirm this to be the case.

III. Simulating the Tangle

It is clear that after the genesis transaction, the growth of $\mathcal{T}$ is uniquely described by the attachment rules, i.e. the tip selection mechanism. The behaviour of this process in the long run is akin to a random graph in a random environment and, like similar models such as diffusion-limited aggregation [15] is simple to state but difficult to analyse mathematically. Its structure however can be gleaned via simulations.

A. NetLogo

NetLogo [16] is a well-known, widely used, cross-platform modelling and simulation environment for complex systems of concurrently interacting agents written on top of the Java Virtual Machine that inherits much of its advanced concurrency and library support [17], thus making it expressive and powerful and customisable. Its main attractiveness comes from its being based on a Logo dialect extended to support agents and modern programming paradigms, well-suited for rapid prototyping of complex scenarios, using "turtles" (agents), "agentsets" as collections of agents that can be customised on the fly, "patches" (the spatial coordinates on which agents sit), "links" (relationships between turtles), and "extensions" – libraries that enhance the core functionality of NetLogo. Of the latter, we use the `nw` extension[7] to analyse the network structure of the Tangle; this simple yet powerful and flexible construction makes it straightforward to include multiple strategies and define functions (`procedures`) and statistics (`reporters`), and the built-in interface and tools, in particular the 2D and 3D views, buttons, switchers, choosers and plots provide an intuitive way to look for structure as the mixture of agent strategies change. Our models of the growth of the Tangle are implemented entirely in NetLogo[8]. We also build a strategy selector menu which can be used to implement different strategies in function of other node internal variables in addition to the cumulative weight.

B. Random Uniform Growth

The simplest version of our NetLogo code, based on the random growth of Subsubsection II-C with instantaneous approvals, makes it easy to generate fast, efficient samples of the Tangle. For example, one can easily scale and visualise in real time up to tens of thousands of nodes being added and confirmed; on a machine with 16GB of RAM and a 2.7GHz multicore i5 processor running MacOS 10.13 or Ubuntu 17.10 this takes about 10 minutes. We define two initial global parameters, `genesis` for the genesis transaction(s), and `lambda` for the average number of incoming tips, which is assumed to be drawn from a Poisson distribution via a reporter function. We also assume that turtles (called nodes by the use of a `breed` keyword) have an internal variable cw that records their cumulative weight. Likewise, directed links (breeded as edges) are used to construct the Tangle structure. Other internal variables are possible and commented in the more complicated models. The procedure `setup-tangle` initialises the Tangle by creating a star network of edges directed from the initial tips to the genesis, and sets their initial cw which, for simplicity, is 2 for the genesis and 1 for the tips. Thereafter, the procedure `grow-tangle` does three things, implemented as further procedure calls: a) it finds tips to attach – i.e. it approves nodes created by the Poisson clock; b) it updates the internal variable cw by incrementing it if a node has an incoming edge; and c) it advances the simulation by one discrete tick. Optionally, it can also update the spatial layout and any visual feedback before the `tick` command is given, for example by colouring nodes with similar cw. Both main procedures above, and the ancillary `approve-nodes`, `update-cw` procedures and `incoming-tips` reporter, take less than 20 lines of terse code. The typical shape of a large random Tangle sample and of its adjacency matrix is shown in Figures 1a and 1b; its depth is described by the layering of cumulative weight, and its width by the average incoming tips λ, which is user-modifiable.

C. Growth Using the MCMC Selection Algorithm

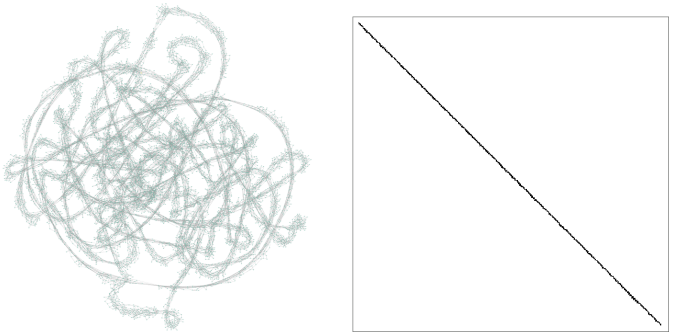
We replicated the Tangle growth strategy using the MCMC-type algorithm, which we simplified for computational tractability by experimenting with the array and network extension primitives `nw:turtles-in-radius`, `nw:path-to`, `nw:turtles-in-reverse-radius` and recursive `ask one-of in-edge-neighbors` calls until

[5]The sparse adjacency matrix for $10,000$ nodes, for example, requires 500MB, and 3GB for 25000, which is still manageable on a 2015 vintage laptop but unwieldy for a large number of sequential writes.

[6]Disregarding for simplicity the proof-of-work nonces and network latency issues, which can be nevertheless be modelled by a compound clock as the average number of revealed tips will be $\lambda \cdot h$ for h the network delay in seconds, which is a constant.

[7]Available in versions 5.3.1 and 6.0.2.

[8]Code and examples are freely available at https://bitbucket.org/mdxmase/iotasim/



(a) (b)

Fig. 1: The "typical" shape of a random Tangle at $\mathcal{T}_{1010}$, with 1039 nodes, $\rho = 3$, $\lambda = 10$, and its adjacency matrix $\mathcal{A}$.

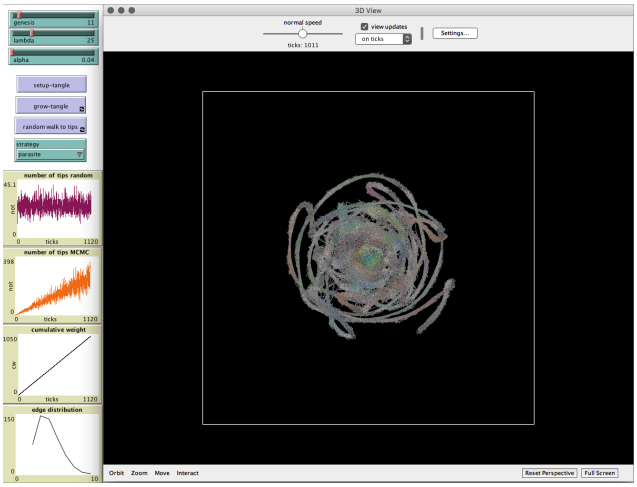


Fig. 2: The NetLogo Interface View with a 3D rendering.

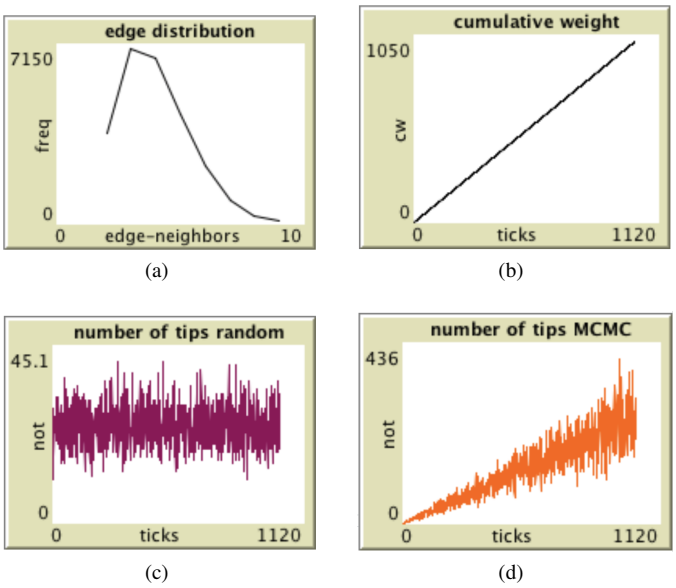
we found a satisfactory solution, and packaged it into a `rw-tips` approval procedure that for each random list of new tips created by the Poisson clock, initialises the chains based on a `num-walkers` parameter and manually backpropagates the Tangle edges to the two selected tips. This is less efficient than random uniform choice of tips, but speeds the computations by a factor of 10 compared to an algorithm that searches the state space at every tick via the transition matrix and a list copy, which is computationally expensive for a large (≥ 100) number of walkers.

D. Initial Findings and Comparisons

The interface setup is shown in Figure 2, which also shows the Netlogo 3D View in action while the Tangle is being simulated using a spring-loaded layout started at the origin, and different colours for cutsets, and two graph and node statistics, namely the number of tips not and the cumulative weight cw. In our exploratory simulations, we replicated the theoretical intuitions of the whitepaper [12] and the simulations of [18] in that the incoming tip process bounds the random fluctuations of the number of tips in $\mathcal{T}(t)$ in a narrow band around a multiplicative constant of λ and λt for the uniform random (Figure 3c) and MCMC tip selection strategies (Figure 3d), respectively; and that the cumulative weight of nodes grows linearly in time after the initial burn-in, as in Figure 3b, while the edge distribution shape (Figure 3a) remains mostly unchanged around an average of 4 as the Tangle grows. We also found that the Tangle samples have a pleasing visual structure. The viewing perspectives of the NetLogo simulator also simplify inspection of nodes when looking for further structure, enabling zooming in and out of the current visualisation, and the `nw` and `filename` extensions provide facilities for saving data as `graphml` and text files for further processing or for using of the simulations as graphical environments for more complex agent strategies.

E. Conjectures

The preliminary analysis of the Tangle $\mathcal{T}$ in NetLogo has also revealed an interesting phenomenon, which is more prominent for some values of α in the range $0.01 - 0.7$. The



(a) (b)

(c) (d)

Fig. 3: Statistics from a random sample of 25256 nodes, stopped at $\mathcal{T}_{1010}$, $\rho = 10$, $\lambda = 25$, $\alpha = 0.01$. (a) Edge distribution: $d(v) := 2 + d_{\mathbf{in}}(v)$. (b) cw of genesis node at $\mathcal{T}_{1010}$, $\rho = 10$, $\lambda = 25$, $\alpha = 0.01$. The other nodes have the same behaviour. (c) Number of tips after $\mathcal{T}_{1010}$ $\rho = 10$, $\lambda = 25$, $\alpha = 0.01$ stays on average around λ. (d) Number of tips after $\mathcal{T}_{1010}$, 25256 nodes. $\rho = 10$, $\lambda = 25$, $\alpha = 2$ drifts linearly in the number of ticks.

DAG structure naturally induces a hierarchy of cutsets in terms of the variable cw, which reveals how the parameter λ really drives the width of the network, acting as a bottleneck when the random clock creates a low number of new nodes.

In addition to what is currently known about the desirability of exponentially biased tip selection strategies as studied in [12], [14], we would like to find out if there is some optimality principle to drive tip selection. We first observe that the architecture of the Tangle Markov Chain can be viewed as a special case of a directed network that has become commonplace in the Deep Learning literature, where each hidden layer is

one state of the chain, and the initial genesis transaction is in a particular sense the higher-level representation of the information stored in the distributed ledger $\mathcal{T}$. Such networks are often randomly initialised and updated. The Tangle of course is not a collection of neurons, but it is very much an artificial information-processing structure. The transition probability in (1) provides a mean-field criterion for signal propagation deep into the network, as recently discovered for DNNs by [19]. An information-theoretical explanation called the *information bottleneck* based on the Bellman optimality principle has been suggested by [20]. If we denote with T a compressed variable, such an algorithm minimises $\min_{p(t|x)} I(X;T) - \beta I(T;Y)$ with $I(X;T)$ and $I(T;Y)$ the *mutual information* between $X;T$ and $T;Y$, respectively, and β a Lagrange multiplier, which provides a growth bound for action by independent agents and walkers on the structure [21]. Just as a DNN is designed to learn how to describe a feature X to predict a function Y and eventually compress X to only hold the information related to Y, in the same vein one might arrive at an efficient, robust representation of information in the distributed ledger that is expressive, relevant information-rich and resistant to attack and propagates information in the network efficiently and instantaneously [22].

F. Multiagent Analysis of Parasite Strategies

Currently, our simulated multi-agent system only implements a mixture of naive parasite strategies such as building an offline linear network and incrementally attaching it to the main Tangle at successive, evenly-spaced points, which tend to have a negligible effect for sufficiently large λ. A further step for the analysis presented here, which we reserve for future work, is to study the evolution of the Tangle information structure under sophisticated attack vectors, which is often found in biological virus attacks. For example, in Section 4.2 of [12], a possible splitting attack scheme towards a Tangle network implementing a MCMC algorithm based on dynamic load-balancing of different branches containing conflicting information is mentioned; these complex strategies could be modelled in our framework by using NetLogo to simulate the environment and implementing agents as malicious nodes using a high-level planning framework based on the Belief-Desire-Intention architecture, either in NetLogo directly or by linking with the Jason development environment as recently done in [23]. Additionally, we consider endowing the nodes of $\mathcal{T}$ with further internal variables, such as smart contract items γ that can be used to deal with trust and distrust, and study contradictory information resolution alongside the approach adopted in [24].

IV. Conclusions

In this paper, we presented the mathematical structure of a block-less, fee-less, distributed ledger technology suitable for the WSN, IoT and CPS which overcomes the limitations of blockchains for these applications, reduces confirmation times to a minimum, and no special computation requirements are imposed, where transactions are nodes of a directed acyclic graph. We also provided a NetLogo simulation environment that makes it easy to simulate and extend analyses of such distributed ledgers. Experimental results performed under this environment confirm known intuitions and available results on the performance of the network and show that DAG-based distributed ledger paradigms hold promise for more complex analyses and applications.

References

[1] D. Gambetta, Ed., *Trust: Making and Breaking Cooperative Relations*. Blackwell, 1990.

[2] M. A. Wooldridge, *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2003.

[3] G. Simmel, *The Philosophy of Money*. Routledge, 1907.

[4] N. R. Kocherlakota, "Money is memory," *Journal of Economic Theory*, vol. 81, no. 2, pp. 232–251, 1998.

[5] J. F. Nash Jr, "Ideal money," *Southern Economic Journal*, vol. 69, no. 1, pp. 4–11, July 2002.

[6] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," October 2008. [Online]. Available: https://bitcoin.org/en/bitcoin-paper

[7] X. Boyen, C. Carr, and T. Haines, "Blockchain-free cryptocurrencies: A framework for truly decentralised fast transactions," https://eprint.iacr.org/2016/871.pdf, 2017.

[8] C. LeMahieu, "RaiBlocks: A feeless distributed cryptocurrency network," December 2017. [Online]. Available: https://raiblocks.net/media/RaiBlocks_Whitepaper__English.pdf

[9] A. Churyumov, "Byteball: A decentralized system for storage and transfer of value," 2015. [Online]. Available: https://byteball.org/Byteball.pdf

[10] (2017) IOTA: A cryptocurrency for the Internet of Things. [Online]. Available: www.iota.org

[11] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "SPECTRE: Serialization of Proof-of-Work Events - confirming transactions via recursive elections," 2017. [Online]. Available: https://eprint.iacr.org/2016/1159.pdf

[12] S. Popov, "The Tangle," (v 1.4), November 2017. [Online]. Available: https://iota.org/IOTA_Whitepaper.pdf

[13] Wikipedia. Adjacency List. [Online]. Available: https://en.wikipedia.org/wiki/Adjacency_list

[14] S. Popov, O. Saa, and E. Finardi. (2017, December) Equilibria in the Tangle. ArXiV. [Online]. Available: https://arxiv.org/abs/1712.05385

[15] Wikipedia. Diffusion-Limited Aggregation. [Online]. Available: https://en.wikipedia.org/wiki/Diffusion-limited_aggregation

[16] U. Wilensky, "NetLogo," Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, Illinois, 1999. [Online]. Available: http://ccl.northwestern.edu/netlogo/docs/

[17] S. Tisue and U. Wilensky, "NetLogo: A simple environment for modeling complexity," Presented at the International Conference on Complex Systems 2014, Boston, May 16–21, 2004.

[18] B. Kusmierz. (2017, November) The first glance at the simulation of the Tangle: discrete model. [Online]. Available: https://iota.org/simulation_tangle-preview.pdf

[19] S. S. Shoenholz, J. Gilmer, S. Ganguli, and J. Sohl-Dickstein, "Deep information propagation," April 2017. [Online]. Available: https://arxiv.org/pdf/1611.01232.pdf

[20] R. Schwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," April 2017. [Online]. Available: https://arxiv.org/pdf/1703.00810.pdf

[21] N. Tishby and D. Polani, "Information theory of decisions and actions," in *Perception-Action Cycle*, ser. Springer Series in Cognitive and Neural Systems, V. Cutsuridis, A. Hussain, and J. Taylor, Eds. Springer, 2010.

[22] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli. (2016, June) Exponential expressivity in deep neural networks throught transient chaos. [Online]. Available: https://arxiv.org/pdf/1606.05340.pdf

[23] W. A. L. Ramirez and M. Fasli, "Integrating NetLogo and Jason: a disaster-rescue simulation," in *Proceedings of 9th Computer Science and Electronic Engineering Conference (CEEC), 27-29 Sept. 2017*. IEEE, 2017, pp. 213–218.

[24] G. Primiero, F. Raimondi, M. Bottone, and J. Tagliabue, "Trust and distrust in contradictory information trasmission," *Applied Network Science*, vol. 2, no. 12, pp. 1–30, 2017.