Utility maximisation in the Coordinator-less IOTA Tangle

Mathilde Jay, Ambre Mollard, Ye Sun, Ruyi Zheng, Isabel Amigo, Alexandre Reiffers-Masson, Santiago Rincón

Utility maximisation in the Coordinator-less IOTA Tangle

Mathilde Jay[1], Ambre Mollard[1], Ye Sun[1], Ruyi Zheng[1], Isabel Amigo[1,2], Alexandre Reiffers-Masson[1,2], and Santiago Ruano Rincón[1,2]

[1] IMT Atlantique, Plouzané, France
first-name.last-name@imt-atlantique.net
[2] Lab-STICC, UMR CNRS 6285, Brest, France,
first-name.last-name@imt-atlantique.fr

Abstract. Getting rid of the Coordinator in the IOTA Tangle is a challenging task, especially regarding the network trustworthiness. Using a customized testbed, we experimentally analyse the functioning of the GoShimmer, IOTA's current implementation of the decentralised Tangle, with respect to specific network performance metrics. We observe that a trade-off exists among such metrics. We thus propose to determine the optimal rate allocation through an optimization problem maximising network performance and user utility. We further propose a distributed and asynchronous scheme to allow nodes to solve such problem.

Keywords: Tangle · Utility maximisation · Distributed rate allocation

1 Introduction

IOTA is an open-source, fee-less, distributed ledger technology and cryptocurrency designed for the Internet of things (IoT). It uses a Direct Acyclic Graph (DAG) to store messages on its ledger, called the Tangle, motivated by a potential higher scalability over blockchain-based alternatives. IOTA does not use miners to validate messages, allowing them to be issued without fees, particularly interesting for micro-transactions. Moreover, the Tangle could be used to store data, and is already used by major actors, such as the Japanese government on NEDO project.

In the initial version of IOTA, the network achieves consensus through a coordinator node. Indeed, IOTA's definition of consensus requires a confirmed message to be referenced (either directly or indirectly) by a signed message issued by the Coordinator (see [11]). This makes the coordinator a single point of failure. To circumvent this problem, a new decentralised version of IOTA, that removes the need of the Coordinator, is being developed along with the GoShimmer node software[3][11]. Being able to get rid of the Coordinator in IOTA is a challenging task, at least from security and system performance points of view (synchronisation, validation rates, etc.). Initial answers have been given in [11],

[3] GoShimmer: https://github.com/iotaledger/goshimmer/

but a lot of questions remain open, especially regarding network management and control considerations.

According to IOTA's terminology, we call *nodes* to the different actors participating in the ledger. Such actors will issue *messages* aiming to have them attached to the distributed ledger graph. In order to issue a message and get it attached, two actions must be done by the node: 1) perform a Proof Of Work (PoW) and 2) validate two messages already present in the graph. Once these two actions performed, the message is added to the graph, and it becomes a *tip*. A tip is thus a message that has not yet been validated by any other message.

Collaboration between the different nodes in the Tangle is essential for an IOTA ledger to be efficient and trustworthy. In addition to validate messages or tips, the nodes need to work together to resolve conflicts via the Fast Probabilistic Consensus (FPC) protocols [10,6]. More details about the Tangle, especially the performance metrics we are paying attention to, can be found in Section 2.1.

In this paper we first experimentally study the behaviour of the decentralised Tangle (Section 2). For that, we have built our own testbed based on publicly available software and study its performance under different scenarios. The main conclusion that can be thrown from these experiments is that a trade-off exists among different performance metrics according to nodes objectives.

In the second part of this paper (Section 3), we focus on finding a good trade-off for the evaluated metrics. We formulate the problem as an optimization one, which allow as to determine the optimal rate allocation. We further propose a distributed and asynchronous scheme to let the nodes reach the identified solution.

1.1 Related Work

Characterising the performance of the Tangle has drawn the attention of several research works, be it through theoretical models, simulations or empirical data. An initial paper [12] introduced the basis for a continuous time model, where several conjectures were presented, in particular regarding several performance metrics. These predictions have been verified through simulations in [8]. A discrete time model has been proposed in [5] where also performance metrics were deduced. In [8], the discrete time model is also validated through simulations and further analytical analysis.

Empirical analysis: The closest work to section 2 is the work of [7] which focus exclusively on the analysis of empirical data in the public Tangle implementation. In this analysis, validation time and other metrics is shown to be more pessimistic than simulations and theoretical modes, while good or bad performance must be stated in regard of a given application. All cited works focused on IOTA's initial version, i.e. the one with a central coordinator. In our case, we focus in the Coordicide [11], the current under development version, which aims to get rid of such coordinator. Such a version is expected to be much more demanding for all performance metrics, since the network is now completely distributed. This constitutes a main difference of our work with current state-of-the-art.

Rate control: Luigi Vigneri and Wolfgang Welz from the IOTA fondation proposed an Adaptive Rate Control Algorithm which adapted the PoW difficulty of a given node from its throughput and its reputation [15,16]. They explain that this algorithm allows every node to issue messages while penalizing spamming actions. The authors claim that fairness is achieved and the issue of mining races is resolved. However, the notion of fairness is not tackled in [15], where authors argue that fairness and other properties must be achieved by properly tuning parameters, without giving further insight on how this tuning can be done neither on fairness measurements. In addition, the approach suggested in [15] does not allow users to add constraints such as a high transmission rate or very low CPU usage. In [16] authors revisit their previous work presenting now a fairness measurement, defined as "the ability of nodes to issue valid messages at a rate independent on their computational capabilities". While this property is doubtless desirable, we claim that a fairness measurement should also take into account the demand of each node, understood as the message rate each node needs to send to the network. We share with these works the objective of rate allocation, while in our problem we take into account the overall utility, understood as network performance and user experience.

Distributed algorithm for network optimisation: A huge amount of work has been dedicated to the design of distributed algorithms for resource allocation and utility maximization. In our case, we will get inspired by the different methods proposed in network utility maximization [17], resource allocation in networks [9], distributed gradient [13] and in general distributed optimization [4,14].

2 Testbed and empirical findings

In our empirical study, we run several scenarios on a network of GoShimmer nodes, where we test different environment conditions. In this paper we focus on studying the impact of two variables: 1) the rate of generated messages, and 2) the PoW difficulty configured for each node.

The messages are generated by a customized spammer plugin. We have modified it to generate messages following a Poisson process, in order to match a common assumption in the literature [12]. Nodes are embedded in a network which is created using the docker-network tool included in the Goshimmer source code. All of the experiments have been run on an Intel Xeon E5-2630 v3 CPU @ 2.4Ghz server. Since all the GoShimmer nodes run on the same virtual server, we assign via docker two different CPU cores to each node, aiming to control and uniform their execution environment. For each experiment, we collect data from a specific GoShimmer plugin that dumps message-related data to disk. The GoShimmer version that we are using can be found in our public git repository as well as the analysed data[4].

[4] Our GoShimmer fork that includes customised plugins: `https://gitlab.imt-atlantique.fr/iota-imt/goshimmer/`

2.1 Relevant performance metrics

According to our use case (network's performance evaluation and utility optimisation), the most relevant metrics to consider are the following:

Throughput: defined in this context as the number of messages that are attached to the Tangle per unit of time. A user is interested in maximizing its throughput according to its current demand and hardware limitations.

Number of tips: tips are messages that have been attached to the Tangle but are not yet validated by any other message. In order to increase reliability of the network, it is desirable that the number of concurrent tips remains slow. In addition, this quantity impacts the user experience since, for same input rate, the higher the number of tips, the longer it takes for one message to get approved.

Finality time: the time elapsed between a message is issued and it is consider as irrevocable. The smaller this metric the fastest a user can have irrevocable messages attached to the Tangle. Determining finality of a message is however not implemented in the current version of GoShimmer (0.3.6), so we are not able to study this metric.

Solidification delay: A message is considered solid by a node when the node knows it and all of its history[11]. An interesting metric describing the performance of the Tangle is the time span elapsed between the following two events: time at which a message is issued (event 1) and its solidification time (event 2). Indeed, the smaller this delay, the fastest nodes are synchronising.

2.2 Scenarios

We focus on homogeneous scenarios –i.e. all nodes have same input parameters–, for this allows to see more clearly the impact of the parameters on network performance. The duration of each experiment is between 5 to 10 minutes. In all cases this was enough to see the metrics converging. These different scenarios allow us to measure the impact of the input rate (messages per minute) and the PoW difficulty, constituting 36 different scenarios. In particular, we can distinguish among scenarios of low, medium and high values for each parameter, where limits for PoW difficulty are given by GoShimmer (between 1 and 20) and for input rate from empirical limits.

2.3 Network performance: Throughput, Solidification Delay and Average number of tips

Fig. 1 shows the obtained results of our different experiments. We report on values about the whole network as seen by one of its nodes. We have checked thus consistency across the different nodes. For some scenarios, consistency was not always achieved, which suggests that nodes are not always able to synchronise.

This was true for different scenarios (be it low, mid or high load), and varied across different runs of same scenario. Such inconsistencies can be seen in some of the reported results (e.g. input rate 3840 mpm for PoW 5). Results obtained still allow us to get the main trends of the metrics, and fully understanding the causes of such anomalies would be the subject of future work.

We can observe, in Fig. 1a that for low scenarios throughput matches the input rate, meaning that nodes are able to issue and send to the network as much as messages as demanded by application. For mid scenarios, this is less and less true since nodes spend more time solving the PoW. For high scenarios, throughput can drop dramatically with respect to input rate. These observations motivate the fact that the PoW can be used to regulate the throughput of each node.

In Fig. 1b, we observe an increase in the number of concurrent tips, when the number of transactions sent to the network or the PoW difficulty increase.

In Fig. 1c, we observe that the solidification delay remains small for mid and low scenarios (the median is less than 0.01 second), while in high scenarios the value increases up (the median is less than 0.05 second). This might have an impact in the increasing number of concurrent tips.

A general observation is that, for high scenarios, the node is not able to catch up with input rate, sending less messages to the network than asked by the application. Solidification delay also increases greatly due to the duration of the PoW. Same observations can be made for the number of concurrent tips.

2.4 Discussion

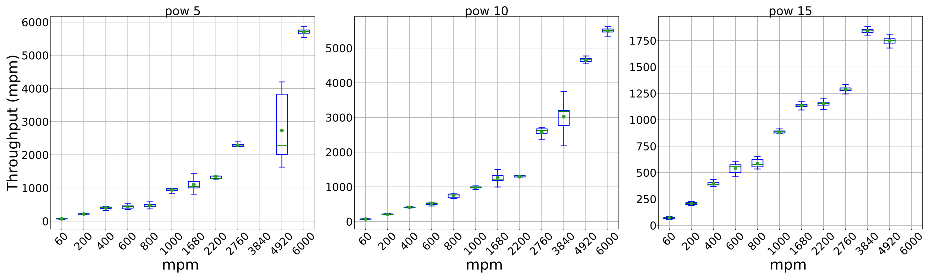
Previous results show that there is a trade-off between the different metrics, and that the best operation point given by the rate as well as PoW difficulty is not trivial to obtain, due to the fact that the metrics depend on these two inputs in a different way. In the next section, we consider that while input rate is determined by the application, PoW difficulty can be pertinently tuned to achieve an optimal trade-off between the different metrics.

3 Network optimisation scheme

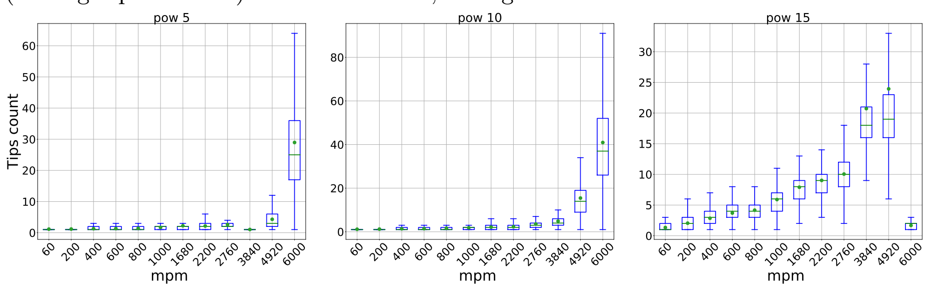
This section is dedicated to the design of a rate allocation scheme which aims to determine the adequate PoW of each node taking into account performance metrics and nodes demands. Our solution is based on an optimisation problem, which we will first define after introducing the system model and assumptions. Then we will design an iterative, distributed and asynchronous scheme which converges to the solution of the optimisation problem. We end the section with numerical results and discussion.

3.1 System model

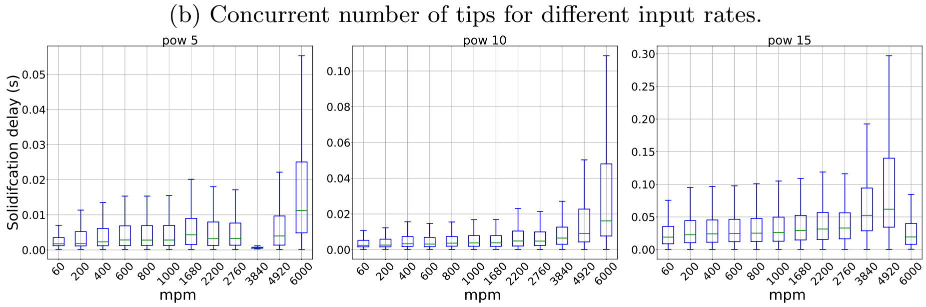
Let $\mathcal{I} := \{1, \ldots, I\}$ be the number of nodes in the system. We consider slotted time where at the beginning of each time slot $n \in \mathcal{N}_+$, a node i generates



(a) Throughput per node (messages per minute) for different per-node input rates (messages per minute). For convenience, the figures are in different scales.



(b) Concurrent number of tips for different input rates.



(c) Solidification delay for different input rates

Fig. 1: Experimental results for the different network performance metrics and scenarios. The number of messages per minute (resp. the metric studied) is represented on the x-axis (resp. on the y-axis). The title of each plot indicates the difficulty of the cryptographic puzzle. The result of each experiment is captured by a box plot.

$0 < \lambda_i < +\infty$ new messages. $\boldsymbol{\lambda} := [\lambda_i]_{1\leq i\leq I}$ is the associated vector. As already proposed in the literature[11,15,16] and in the current GoShimmer implementation, we consider that each node has to solve a cryptographic puzzle before sending a message to the network. We assume that if a message is not able to solve the cryptographic puzzle in $N \in \mathcal{N}_+$ time slots, then the message is rejected by the system. The success probability to solve a puzzle for any message sent by node i, during time slot n, is denoted by α_i. This assumption can be seen as the discrete version of the assumption that the time to solve a cryptographic puzzle follows an exponential law, as suggested in [1,2]. We assume that $\alpha_i \in [\underline{\alpha}_i, \overline{\alpha}_i]$ with $0 < \underline{\alpha}_i < \overline{\alpha}_i < 1$. We denote by $\boldsymbol{\alpha} := [\alpha_i]_{1 \leq i \leq I}$ the associated vector. Let x_i^n be the number of messages sent by node i in the network at the end of time slot n. These messages were still trying to solve the puzzle at the beginning of the time slot n. We call this quantity the *throughput* of node i at time slot n. The expectation of x_i^n, denoted by $\overline{x}_i := E[x_i^n]$ can be computed explicitly:

Lemma 1. *For all n, $n' \geq N$, $E[x_i^n] = E[x_i^{n'}]$ and*

$$\overline{x}_i(\alpha_i) := \overline{x}_i^N = \lambda_i \left(1 - (1 - \alpha_i)^N\right). \tag{1}$$

Observe that $\overline{x}_i(\alpha_i)$ is a strictly increasing concave function of α_i.

Proof. First note that the number of messages, from node i, that are currently solving a dedicated cryptographic puzzle at time slot n are only depending on the messages generated from the beginning of time slot $n - N + 1$ up to the end of time slot n. Moreover, due to the fact that λ_i and α_i are not varying over time, we have that $E[x_i^n] = E[x_i^{n'}]$ for all $n, n' > N$.

Let $z_i^{m,k} \in \{0, 1\}$ be a random variable which is equal to 1 when message m from node i sent at time $k \in \{n - N + 1, \dots, n\}$ has been able to solve the cryptographic puzzle during time slot n. Otherwise $z_i^{m,k}$ is equal to 0. The probability that $z_i^{m,k} = 1$ is equal to $(1 - \alpha_i)^{n-k}\alpha_i$. We have $E[x_i^n]$ which is equal to:

$$E[\sum_{k=n-N+1}^{n} \sum_{m=1}^{\lambda_i} z_i^{m,k}] = E[\sum_{k=1}^{N} \sum_{m=1}^{\lambda_i} z_i^{m,k}] = \sum_{k=1}^{N} \sum_{m=1}^{\lambda_i} E[z_i^{m,k}] = \lambda_i \left(1 - (1 - \alpha_i)^N\right). \blacksquare$$

If we assume that the α_i is changing every N time slots, then this lemma is true for every $n = kN$ with $k \in \mathcal{N}$.

3.2 Modelling network performance metrics

A first metric to observer is the average number of concurrent tips. As mentioned in Section 2, this is indeed an interesting metric to *minimize* as the less number of tips, the earlier a message can be validated. We assume that the average number of tips is a linear increasing function of the total throughput $\sum_{i=1}^{I} \overline{x}_i(\alpha_i)$. In particular, this agrees with our experimental findings as well as with state-of-the-art models and simulations [5,8,12]. We have thus that the mean number of tips in the Tangle at time slot n is given by $g(\sum_{i=1}^{I} \overline{x}_i(\alpha_i)) = c_1 \sum_{i=1}^{I} \overline{x}_i(\alpha_i)$, where c_1 is a constant depending on modelling assumptions.

Secondly, we consider each node's throughput as a metric to be *maximized*. We assume that every node i, at instant n, is interested in maximizing a concave (to capture the diminishing returns effect) increasing and differentiable function $U_i(\overline{x}_i(\alpha_i))$ of the throughput rate $\overline{x}_i(\alpha_i)$. Note that $U_i(\overline{x}_i(\alpha_i))$ is concave in α_i as long as $U_i(\cdot)$ is an increasing concave function. We could assume that $U_i(\overline{x}_i(\alpha_i)) = w_i \log(\overline{x}_i(\alpha_i))$.

Finally, as aforementioned, finality time is an interesting metric to be optimised, however it is not yet available neither experimentally nor theoretically (no models exist in the literature). We thus consider an alternative metric, the average *confirmation time*, i.e. the average time elapsed between the following two events: the message is issued by the node (event 1) and the message is no longer a tip (event 2). Such metric is important in terms of quality of experience, since it considers the pace at which a user can add messages to the Tangle.

We assume that the confirmation time is a decreasing function of the total throughput which agrees with theoretical models (see [5]). The confirmation time is assumed to be given by a non linear decreasing convex function $h(\sum_{i=1}^{I} \overline{x}_i(\alpha_i))$. Note that this function is convex in the vector $\boldsymbol{\alpha}$ as the composition of a concave function (sum of concave functions) with a non-increasing convex function over an univariate domain. In particular, we can suppose that $h(\sum_{i=1}^{I} \overline{x}_i(\alpha_i)) = \frac{1}{\sum_{i=1}^{I} \overline{x}_i(\alpha_i)}$.

Note we are not considering solidification time as an objective to be optimised. Indeed, though this is an interesting metric, experimental results have shown not to be a stable one, and no models can be safely extracted from data. Considering such metric is thus left for future work.

3.3 Optimization problem

Without loss of generality we define the utility maximization problem as a minimization cost problem. The instantaneous cost function, $J(\boldsymbol{\alpha})$, is the weighted sum of the quantities introduced in previous sub-section, described as follows:

$$\begin{aligned} J(\boldsymbol{\alpha}) &:= \lim_{n\to+\infty} \frac{1}{n} \sum_{k=0}^{n} c_1 \underbrace{\sum_{i=1}^{I} \overline{x}_i^n(\alpha_i)}_{\text{Inst. average tip count}} + \underbrace{h(\sum_{i=1}^{I} \overline{x}_i^n(\alpha_i))}_{\text{Inst. confirm. time}} - \underbrace{\sum_{i=1}^{I} U_i(\overline{x}_i^n(\alpha_i))}_{\text{Network Utility}} \\ &= c_1 \sum_{i=1}^{I} \overline{x}_i(\alpha_i) + h(\sum_{i=1}^{I} \overline{x}_i(\alpha_i)) - \sum_{i=1}^{I} U_i(\overline{x}_i(\alpha_i)). \end{aligned}$$

We thus propose to solve the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & J(\boldsymbol{\alpha}), \\ \text{s.t.} \quad & \alpha_i \in [\underline{\alpha}, \overline{\alpha}],\ \forall i \in \mathcal{I}. \end{aligned} \tag{2}$$

This optimization problem can be solved in two steps. Indeed, due to the fact that for all $i \in \mathcal{I}$, $\overline{x}_i(\alpha_i)$ is strictly increasing in $\alpha_i \in [\underline{\alpha}, \overline{\alpha}]$, we can:

– Firstly, solve the following optimization problem:

$$\begin{aligned} \min_{\overline{\boldsymbol{x}}} \quad & c_1 \sum_{i=1}^{I} \overline{x}_i + h(\sum_{i=1}^{I} \overline{x}_i) - \sum_{i=1}^{I} U_i(\overline{x}_i), \\ \text{s.t.} \quad & \overline{x}_i \in [\overline{x}_i(\underline{\alpha}), \overline{x}_i(\overline{\alpha})],\ \forall i \in \mathcal{I}, \end{aligned} \tag{3}$$

where $\overline{\boldsymbol{x}} := [\overline{x}_i]_{1\le i\le I}$. This problem is a convex optimization problem.

– Secondly, if we denote by $\overline{\boldsymbol{x}}^*$ the solution of the optimization problem, then the optimal α_i^* is equal to $x_i^{-1}(\overline{x}_i^*) = 1 - (1 - \frac{\overline{x}_i^*}{\lambda_i})^{1/N}$, for all $i \in \mathcal{I}$.

In the next lemma, following the theory of convex optimization we derive an explicit solution of (3) for specific functions.

Lemma 2. *If for every $i \in \mathcal{I}$, $U_i(\overline{x}_i) = w_i \log(\overline{x}_i)$, $h(\sum_{i=1}^I \overline{x}_i) = \frac{c_2}{\sum_{i=1}^I \overline{x}_i}$, with $w_i > 0$ and $c_2 > 0$ and if $\overline{x}_i^* \in (\overline{x}_i(\underline{\alpha}), \overline{x}_i(\overline{\alpha}))$ for all $i \in \mathcal{I}$, then*

$$\overline{x}_i^* = w_i \frac{\sqrt{(\sum_{j=1}^I w_i)^2 + 4c_1c_2} + \sum_{j=1}^I w_i}{2c_1 \sum_{j=1}^I w_i}.$$

Proof. Let us assume that $\overline{x}_i^* \in (\overline{x}_i(\underline{\alpha}), \overline{x}_i(\overline{\alpha}))$ for all $i \in \mathcal{I}$, then the first order optimality condition is equal to:

$$\frac{w_i}{\overline{x}_i^*} - c_1 + \frac{c_2}{(\sum_{j=1}^I \overline{x}_j^*)^2} = 0,\ \forall i \in \mathcal{I},$$

which is equivalent to $\overline{x}_i^* = \frac{w_i}{c_1 - \frac{c_2}{(\sum_{j=1}^I \overline{x}_j^*)^2}}$, for all $i \in \mathcal{I}$. By taking the sum over i, we obtain:

$$\sum_{j=1}^I \overline{x}_j^* = \frac{\sum_{j=1}^I w_i}{c_1 - \frac{c_2}{(\sum_{j=1}^I \overline{x}_j^*)^2}} \Leftrightarrow (\sum_{j=1}^I \overline{x}_j^*)^2 c_1 - (\sum_{j=1}^I \overline{x}_j^*) \sum_{j=1}^I w_i - c_2 = 0$$

$$\Rightarrow \sum_{j=1}^I \overline{x}_j^* = (2c_1)^{-1} \left(\sum_{j=1}^I w_i + \sqrt{(\sum_{j=1}^I w_i)^2 + 4c_1c_2} \right).$$

The last implication is coming from the fact $\sum_{j=1}^I \overline{x}_j^* > 0$ and the fact that the solution $\frac{\sum_{j=1}^I w_i - \sqrt{(\sum_{j=1}^I w_i)^2 + 4c_1c_2}}{2c_1}$ is always negative. We can now conclude our proof by plugging $\sum_{j=1}^I \overline{x}_j^*$ into $\overline{x}_i^* = \frac{w_i}{c_1 - \frac{c_2}{(\sum_{j=1}^I \overline{x}_j^*)^2}}$. ■

3.4 Distributed and asynchronous algorithm

We now describe an asynchronous and distributed algorithm which will converge to the optimal solution of (3).

We assume that every node updates their throughput at each kN time slot, for every $k \in \mathbb{N}$. We denote by $\overline{x}_i(k)$ the throughput adopted by node i, during time slots $\{kN, \ldots, (k+1)N - 1\}$. Only node i observes $\overline{x}_i(k)$ at every k. Let $Y(k)$ be the random subset of $\mathcal{I}$ indicating the subset of nodes which update their throughput at the beginning of the time slot kN. We assume that when a node i is active at time slot kN, it observes the total throughput $\sum_{j=1}^I \overline{x}_j(k)$. We also need to define the step sizes $\{a(k)\}, \{b(k)\} \in (0,1)$ such that: (1) $\sum_k a(k) = \sum_k b(k) = \infty$, (2) $\sum_k a^2(k) + b^2(k) < \infty$ and finally (3) $\lim_{k\to\infty} \frac{a(k)}{b(k)} = 0$. For instance, the functions $b(k) = \frac{1}{k^{2/3}}$ and $a(k) = \frac{1}{k}$ satisfy the above mentioned conditions. We explain the importance of such assumptions over the step-size later on, when we discuss convergence. Let us first describe our algorithm.

Local iteration of node i

Initialization: Set $\overline{x}_i(0)$.

When $i \in Y(k+1)$:
(1) *Clock update step:* Node i observes $k+1$.
(2) *Aggregation step:* Node i observes $\sum_{j=1}^{I} \overline{x}_j(k)$ and updates:

$$y_i(k+1) = y_i(k) + b(k+1)\left(\sum_{j=1}^{I} \overline{x}_j(k) - y_i(k)\right).$$

(3) *Gradient ascent step:* Update:

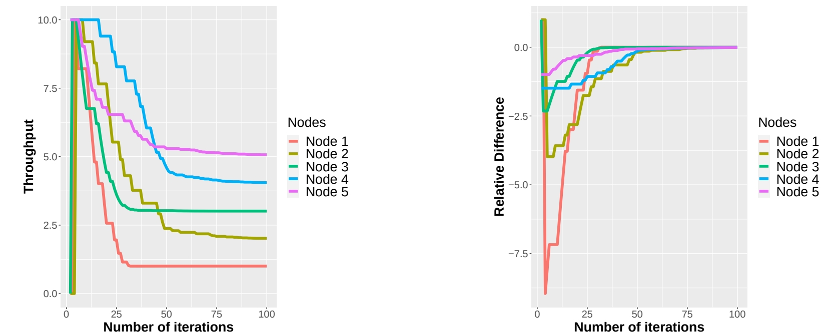
$$\overline{x}_i(k+1) = \left[\overline{x}_i(k) - a(k)\left(c_1\overline{x}_i(k) + h'(y_i(k)) - U_i'(\overline{x}_i(k))\right)\right]_{\overline{x}_i(\underline{\alpha})}^{\overline{x}_i(\overline{\alpha})},$$

where $[x]_a^b = \max\{\min\{x, b\}, a\}$.
Node i adopts throughput $\overline{x}_i(k+1)$ during the instants $\{(k+1)N, \ldots, (k+2)N-1\}$.

When $i \notin Y(k)$:$y_i(k+1) = y_i(k)$ and $\overline{x}_i(k+1) = \overline{x}_i(k)$. So node i adopts the strategy $\overline{x}_i(k+1) = \overline{x}_i(k)$ during the instants $\{k+1)N, \ldots, (k+2)N-1\}$.

Convergence: The mathematical proof of the convergence of our algorithm is out of the scope of this paper. We will however briefly mention the different main ideas. Our algorithm is nothing more than an asynchronous stochastic gradient descent, with biased but consistent estimator of the gradient (see 10.2 in [3]). To prove the convergence almost surely of such scheme one needs to use the theory of stochastic approximation and more specifically two-time scale stochastic approximations and asynchronous stochastic approximations (see ch. 6 and 7 in [3]). The assumptions regarding the time-steps are standard and will ensure that every node i has a stable estimate of the total throughput $(\sum_{j=1}^{I} \overline{x}_j(k))$, decoupled of the gradient update $(\overline{x}_j(k+1))$.

The behaviour of our distributed/asynchronous scheme is illustrated in Fig. 2. We have 5 nodes. At each iteration, 2 nodes are randomly selected and perform an update of their throughput by following our scheme. The parameters are set to $w_i = i$, $c_1 = 1$, $c_2 = 1$ and $\lambda = 10$. We observe that our scheme converges to the optimal throughput in less that 75 iterations (see Fig. 2a and 2b).



(a) Evolution of the throughputs. (b) Evolution of relative difference.

Fig. 2: Convergence of proposed scheme. Parameters are set to $w_i = i$, $c_1 = 1$, $c_2 = 1$ and $\lambda = 10$. Fig. 2a depicts the evolution of throughputs. The evolution of relative difference between the optimal throughput and the one generated by the asynchronous and distributed scheme is depicted in Fig. 2b.

4 Conclusion

We have focused on the study of the Tangle as an example of DAG-based ledgers. We have built a testbed and evaluated network performance under different input conditions, concluding that network's health and user experience depend on these input values, while a trade-off exists among the different considered metrics. A smart control is then needed in order to properly set such parameters.

We have thus defined a network optimisation problem which derives the optimal throughput for every node. For particular functions, we have provided a closed form solution. We have designed a distributed and asynchronous algorithm which converges to the optimum of our network optimisation problem.

This algorithm can be extended to a more complex set-up, e.g. noisy observations or exact shape of the function unknown by the nodes. This along with considering further constraints are among the directions of our future work.

Acknowledgments

Authors would like to thank the IOTA Foundation Team for their kind and prompt assistance, especially Angelo Capossele, who has helped us to understand a little bit the GoShimmer's guts.

References

1. Altman, E., Menasché, D., Reiffers-Masson, A., Datar, M., Dhamal, S., Touati, C., El-Azouzi, R.: Blockchain competition between miners: a game theoretic perspective. Frontiers in Blockchain **2**, 26 (2020)

2. Altman, E., Reiffers, A., Menasche, D.S., Datar, M., Dhamal, S., Touati, C.: Mining competition in a multi-cryptocurrency ecosystem at the network edge: A congestion game approach. ACM SIGMETRICS Performance Evaluation Review **46**(3), 114–117 (2019)
3. Borkar, V.S.: Stochastic approximation: a dynamical systems viewpoint, vol. 48. Springer (2009)
4. Borkar, V.S., Shah, S.M.: Distributed algorithms: Tsitsiklis and beyond. In: 2018 Information Theory and Applications Workshop (ITA). pp. 1–9. IEEE (2018)
5. Bramas, Q.: The Stability and the Security of the Tangle (Apr 2018), https://hal.archives-ouvertes.fr/hal-01716111, working paper or preprint
6. Capossele, A., Mueller, S., Penzkofer, A.: Robustness and efficiency of leaderless probabilistic consensus protocols within byzantine infrastructures. arXiv preprint arXiv:1911.08787 (2019)
7. Guo, F., Xiao, X., Hecker, A., Dustdar, S.: Characterizing iota tangle with empirical data. In: GLOBECOM 2020 - 2020 IEEE Global Communications Conference. pp. 1–6 (2020). https://doi.org/10.1109/GLOBECOM42002.2020.9322220
8. Kusmierz, B., Staupe, P., Gal, A.: Extracting tangle properties in continuous time via large-scale simulations. technical report., https://www.iota.org/foundation/research-papers
9. Li, C., Yu, X., Huang, T., He, X.: Distributed optimal consensus over resource allocation network and its application to dynamical economic dispatch. IEEE Transactions on Neural Networks and Learning Systems **29**(6), 2407–2418 (2017)
10. Popov, S., Buchanan, W.J.: Fpc-bi: Fast probabilistic consensus within byzantine infrastructures. arXiv preprint arXiv:1905.10895 (2019)
11. Popov, S., Moog, H., Camargo, D., Capossele, A., Dimitrov, V., Gal, A., Greve, A., Kusmierz, B., Mueller, S., Penzkofer, A., et al.: The coordicide (2020), https://files.iota.org/papers/20200120_Coordicide_WP.pdf
12. Popov, S., Saa, O., Finardi, P.: Equilibria in the tangle. Computers & Industrial Engineering **136**, 160–172 (2019). https://doi.org/10.1016/j.cie.2019.07.025
13. Pu, S., Shi, W., Xu, J., Nedic, A.: Push-pull gradient methods for distributed optimization in networks. IEEE Transactions on Automatic Control (2020)
14. Shah, S.M., Borkar, V.S.: Distributed stochastic approximation with local projections. SIAM Journal on Optimization **28**(4), 3375–3401 (2018)
15. Vigneri, L., Welz, W., Gal, A., Dimitrov, V.: Achieving fairness in the tangle through an adaptive rate control algorithm. In: 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). pp. 146–148 (2019). https://doi.org/10.1109/BLOC.2019.8751358
16. Vigneri, L., Welz, W.: On the fairness of distributed ledger technologies for the internet of things. In: 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). pp. 1–3. IEEE (2019). https://doi.org/10.1109/ICBC48266.2020.9169465
17. Wei, E., Ozdaglar, A., Jadbabaie, A.: A distributed newton method for network utility maximization–i: Algorithm. IEEE Transactions on Automatic Control **58**(9), 2162–2175 (2013)