Fairness and Efficiency in DAG-based Cryptocurrencies*

Georgios Birmpas[1], Elias Koutsoupias[1], Philip Lazos[2], and Francisco J. Marmolejo-Cossío[1]

[1] University of Oxford
[2] Sapienza University of Rome

Abstract. Bitcoin is a decentralised digital currency that serves as an alternative to existing transaction systems based on an external central authority for security. Although Bitcoin has many desirable properties, one of its fundamental shortcomings is its inability to process transactions at high rates. To address this challenge, many subsequent protocols either modify the rules of block acceptance (longest chain rule) and reward, or alter the graphical structure of the public ledger from a tree to a directed acyclic graph (DAG).
Motivated by these approaches, we introduce a new general framework that captures ledger growth for a large class of DAG-based implementations. With this in hand, and by assuming *honest* miner behaviour, we (experimentally) explore how different DAG-based protocols perform in terms of *fairness*, as well as *efficiency*. To do so, we isolate different parameters of the network (such as k, the number of pointers to previous blocks) and study their effect on those performance metrics.
Our results demonstrate how the DAG-based ledger protocols described by our framework cope with a high transaction load. More specifically, we show that even in a scenario where every miner on the system is honest in terms of when they publish blocks, what they point to, and what transactions each block contains, fairness and efficiency of this kind of ledgers can break down at specific hash rates if miners have differing levels of connectivity to the P2P network sustaining the protocol.[3]

1 Introduction

Bitcoin and many other decentralised digital currencies maintain a public ledger via distributed consensus algorithms implemented using blockchain data structures. End users of the currency post transactions to the P2P network sustaining

* Georgios Birmpas was supported by the ERC Advanced Grants 321171 (ALGAME) and 639945 (ACCORD). Philip Lazos is supported by the ERC Advanced Grant 788893 (AMDROMA) and the MIUR PRIN project ALGADIMAR. Francisco J. Marmolejo-Cossío was supported by the Mexican National Council of Science and Technology (CONACyT). We would also like to thank Elizabeth Smith for fruitful discussions during the preparation of this work.

[3] The full version of this paper can be found in [2].

the protocol and said transactions are bundled into blocks by *miners*: agents tasked with the upkeep of the ledger. With respect to Bitcoin, the prescribed *longest chain rule* dictates that miners must bundle pending transactions into a block that also includes a single hash pointer to the end of the longest chain seen by the miner in their local view of the ledger. Furthermore, in order for a block to be valid, its hash must lie below a dynamically adjusted threshold. Hence, miners must expend computational resources to find valid blocks. Due to this *Proof-of-Work* structure, if all miners follow the protocol, the number of blocks they contribute to the blockchain is proportional to the computational resources they dedicate to the protocol, i.e. their hash power. In addition, miners are incentivised to follow the protocol via judicious incentive engineering through block rewards. This latter point also implies that miners earn block reward proportional to their hash power, thus making Bitcoin a *fair* protocol.

As mentioned before, Bitcoin dynamically adjusts its target hash for valid blocks so that the totality of all miners active in the protocol find a block every ten minutes on average. This feature of the protocol makes consensus more robust, as this time-scale is much larger than the time it takes for a block to propagate on the P2P network supporting Bitcoin. However, since the size of blocks is limited, Bitcoin inherently suffers from a scalability problem. Thus in spite of Bitcoin being strategy-proof and fair, it suffers in its *efficiency*: which we define as the expected ratio of the number of valid transactions in the ledger to the number of all transactions posted in the P2P network. On the other hand, simply decreasing confirmation times and demanding higher transaction throughput by either increasing the overall block creation rate or block size can also affect these very properties of the protocol. For instance, delays in the P2P network may cause miners to have different views of the ledger, which can in turn directly make achieving a consensus more difficult, or lead miners to be strategic when they would have otherwise acted honestly. Ultimately, it seems that Bitcoin fundamentally strikes a delicate balance between being strategy-proof and fair at the cost of efficiency.

There have been many attempts to cope with Bitcoin's inherent throughput limitations, with [16, 21–23] being some notable examples. All of these papers focus on how security can be maintained when the throughput is increased and follow the common direction of either modifying Bitcoin's longest chain rule or implementing a different graphical structure underlying the ledger.

In GHOST [21] an alternative consensus rule is proposed to the longest chain of Bitcoin, focusing on creating a new protocol that maintains security guarantees even when faced with high transaction loads. In this setting, GHOST takes into account the fact that forks are more likely to be produced when the underlying ledger still takes the form of a tree, as with Bitcoin. More specifically, when deciding what a newly mined block should point to, GHOST no longer myopically points to the head of the longest chain, but rather starts from the genesis block and at each fork, chooses the branch of the fork that leads to the heaviest subtree in the ledger until reaching a leaf to point to. In this way, blocks that are off the main chain can still contribute to the final consensus, which arguably maintains a degree of robustness to strategic mining while coping with high throughput better than Bitcoin.

In [22, 23] protocols SPECTRE and PHANTOM are proposed, with ledger structures in the form of directed acyclic graphs (DAG). The protocols in both of these implementations suggest that every newly created block has to point to every available (visible) leaf in the ledger. In that way every created block will eventually become part of the consensus, and the security of the system remains unaffected by forks that will be produced due to high throughput, since they will in turn be part of the ledger. A possible advantage is that the system can become more resilient to attacks that focus on increasing the block rewards of a miner. On the other hand, ordering the transactions and preventing other types of strategic behaviour becomes more complicated.

Motivated by these ideas, we design a new theoretical framework that captures a large family of DAG-based ledger implementations (including those mentioned in previous paragraphs). We achieve this by introducing a parametric model which lets us adjust the number of blocks each newly created block can point to, the block attributes a miner takes into account when choosing what blocks to point to, and the number of transactions a block can store. Finally, we describe a theoretical framework for ledger growth in these DAG-based models, along with a novel simplification for extrapolating valid transactions from a ledger under the assumption that all miners are honest. With this in hand, we are able to answer how our family of DAG-based ledgers copes with the high transaction loads they are intended to tackle. Indeed, our results are structural in nature, for we show how fairness and efficiency suffer from high transaction rates in spite of all agents behaving honestly in a given DAG-based ledger.

We want to mention at this point that we are mostly interested on parameter k, the number of pointers to previous blocks. In contrast to the existing literature regarding DAG-based protocols which assumes that k is conditioned to other parameters of the system (i.e. informational parameters q in our model), something crucial for the security of this kind of protocols, we choose to study this parameter unconditioned for reasons that we will explain shortly.

Our Results. We provide a parametric model that tries to capture a large family of DAG-based ledgers and we make an attempt to quantify what is the effect of adjusting the different parameters of our model on *fairness* and *efficiency*. As we already mentioned, an important parameter for our model is k, for which we assume that is specified by the protocol, fixed and independent of the other parameters (i.e *informational* parameters) that we will eventually introduce in our model. Although this comes in contrast with most of the existing literature where k depends on the informational parameters of the system, the reason behind this choice is twofold: We want to study the contribution of parameter k to the protocol in terms of *fairness* and *efficiency*, while in addition we desire to explore what would happen in a situation where the optimal value of k, the value that does not produce orphaned blocks and thus makes the protocol inherently fair, cannot be selected (i.e. when it is huge). Although this approach may lead to loss of *security* for the protocol (since the chosen k may not be the optimal one), we want to explore how this parameter affects *fairness* and *efficiency* under the assumption of *honest* mining.

In this line of thought, our simulations allow us to show how specific transaction load regimes affect the efficiency of different protocols in our class of DAG-based ledgers. Furthermore, we show that in almost all transaction load regimes *fairness* is affected and exhibits a complicated relationship with respect to agent connectivity in the underlying P2P network. Our results are exploratory in nature, fixing most aspects of the network (assuming a simple, or worse case, setting if possible) and modulating specific parameters to study their effect on fairness and efficiency.

We are interested in exploring the behaviour of the protocol under several value choices of k and q. Some highlights of our results, that we consider both interesting and surprising, are the following: 1) Although we assume honest behaviour from the miners, it is interesting to explore the performance for choices of k and q that increase the security of the system (i.e. choices of values where both the number of the pointers as well as the information that the miners have for the network is high). In our simulations the safest such zone is for $k \geq 2$ and $q = 0.2$. In particular, for smaller values of k we observe that fairness is compromised leading to interesting mining behaviour: the gains of small miners are generally increasing in their q_i, whereas for larger miners they *decrease*: this is because small miners care more about making sure their few blocks are retained, while large miners appear to act 'selfishly' by mining in parallel to the others' blocks, not by malice but by ignorance. 2) Leaving the security of the system aside, another region that we find interesting is the one where q can vary from 0.0001 up to 1 and for which we observe that there is a huge increase in the efficiency of the system as k increases from 1 to 2. On the other hand, something that seems surprising is that by increasing the value of parameter k to 3 or even to ∞ seems that does not provide a significant added benefit to the efficiency of the system. This quite interestingly implies that we can achieve efficiency guarantees even if we do not choose the optimal value for k (since this behaviour is the same for a variety of values of q).

Related Work. Bitcoin was introduced in Nakamoto's landmark white paper [18] as a decentralised digital currency. Since its inception many researchers have studied several aspects of the protocol, i.e. its security and susceptibility to different types of attacks [1, 7, 8, 10, 17, 19], how it behaves under a game-theoretic perspective [4, 14, 15] and how its scalability and inherent transaction throughput issues can be improved. Since the latter is the most related to our work, we give a more detailed exposition in the paragraphs that follow. Before we proceed, we also want to refer the reader to [3, 24] for some extensive surveys which provide a good view of the research and challenges that exist in the area.

Sompolinksy and Zohar [21], study the scalability of Bitcoin, analysing at the same time the security of the protocol when the delays in the network are not negligible. More specifically, they build on the results of Decker and Wattenhofer [5] and explore the limits of the amount of transactions that can be processed under the protocol, while also studying how transaction waiting times can be optimised when there is also a security constraint. In the same work, the Greedy Heaviest-Observed Sub-Tree chain (GHOST) is also presented as a modified version of the Bitcoin protocol selection rule, and as a way of obtaining a more scalable system with higher security guarantees. It is interesting to mention that many existing cryptocurrencies currently use variations of the GHOST rule, with Ethereum [6] and Bitcoin-NG [9] being some notable examples. The authors argue that under this rule, the possible delays of the network cannot affect the security of the protocol even if the designer allows high creation rates of large-sized blocks and thus a high transaction throughput.

Subsequently, Kiayias and Panagiotakos [13] further study the GHOST protocol and provide a general framework for security analysis that focuses on protocols with a tree structure. They expand upon the analysis of [21] and follow a direction similar to the one presented in the work of Garay et al. [10], which only studies chain structures and cannot be directly implemented in the setting of GHOST. We would like to point out that in [10] Garay et al. also provide an extended analysis of their framework for the partially synchronous model under the existence of bounded delays in the underlying P2P network of the protocol.

Lewenberg et al. [16] propose the structure of a DAG, instead of a tree, as a way of dealing with high block creation rates and blocks of large size. Building on this idea, the same authors in [23] present SPECTRE, a new PoW-based protocol that follows the DAG-structure, and is both, scalable and secure. More specifically, they argue that SPECTRE can cope with high throughput of transactions while also maintaining fast confirmation times. They also analyse its security by studying several types of attacks. Part of the contribution of the paper is also introducing a way to (partially) order created blocks via a voting rule among existing blocks, which also contributes to the security of the protocol. SPECTRE has drawn the attention of many researchers after its introduction and we refer the reader to [11, 12, 20, 22] for some indicative related works.

2 DAG-based Ledgers

In this section we will describe a family of decentralised consensus algorithms for public ledgers that generalise Bitcoin and SPECTRE. In what follows, we assume that there are n strategic miners $m_1, ..., m_n$ with hash powers $h_1, ..., h_n$ respectively. When a given block is found globally by the protocol, h_i represents the probability that this block belongs to m_i. We will be studying DAG-based ledger implementations. Formally, these ledgers are such that blocks and their pointers induce a directed acyclic graph with blocks as nodes and pointers as edges. The maximum out-degree of a block, k is specified by the protocol and is in the range $1 \leq k \leq \infty$. Thus it is straightforward to see that Bitcoin for example, is a DAG-based ledger where the DAG is in fact a tree (with $k = 1$). Finally, since blocks have bounded size, we define $1 \leq \eta < \infty$ to be the maximum number of transactions a block can store.

As mentioned in the introduction, we are primarily interested in studying issues of fairness and ledger efficiency in DAG-based protocols catered to a high throughput regime. We recall that a protocol is fair if a miner can expect to see a block reward proportional to their hash power, and that a protocol efficiency is the ratio of all valid transactions to all transactions broadcast over the P2P network. In this setting, and under the assumption of a discrete time horizon, transactions and blocks that are propagated by users in the P2P network may take multiple *turns* (the time it takes for the entire system to find a block) before they are seen by certain agents within the system. For this reason, miners only see a portion of the entire block DAG produced by a decentralised protocol as well as a portion of all transactions propagated by all end users of the ledger.

In actuality, transactions that are posted to the P2P network of digital currencies directly depend on other transactions. For this reason, we also model the set of transactions that end users generate as a DAG. Furthermore, the structure of the transaction DAG itself has important implications for how transactions are packed in blocks for any DAG-based ledger. For example, if the transaction DAG is a path, and we are considering SPECTRE as our DAG-based protocol, it is easy to see that transactions will only be packed proportional to the deepest node of the block DAG, which in the high throughput regime can grow at a much slower rate than that at which transactions are generated. At the other extreme, if the transaction DAG only consists of isolated nodes, then any block can contain any transaction, and the efficiency of SPECTRE is thus constrained by what transactions miners see rather than the structure of the block DAG.

Ultimately, in addition to having computational power, a miner also has informational power, which encapsulates how connected they are to the P2P network and consequently, how much of each of the aforementioned DAGs they see at a given time. We model the informational parameter of an arbitrary miner m_i as a parameter $q_i \in [0, 1]$. As q_i approaches 1, m_i is likely to see the entirety of both DAGs, whereas as q_i approaches 0, m_i is likely to only see the blocks he mines and transactions he creates.

2.1 Ledger Growth Preliminaries

We begin by setting some preliminary notation about graphs that it will be used in several parts as we define the model. Let $\mathcal{G}$ be the set of all finite directed graphs. For $G \in \mathcal{G}$, $V(G)$ and $E(G) \subseteq V(G)^2$ are the set of vertices and directed edges of G respectively. Furthermore, for a tuple $x = (x_i)_{i=1}^n$, we let $\pi_i(x) = x_i$ be the projection onto the i-th coordinate. Finally, we define the *closure* of a subset $X \subseteq V(G)$ of vertices, which will be needed in order to describe how a miner perceives the current state of the network.

Definition 1 (Closure). *Suppose that $G \in \mathcal{G}$, and let $X \subseteq V(G)$ be a subset of vertices. We denote the closure of X in G by $\Gamma(X \mid G)$ and define it as the subgraph induced by all vertices reachable from X via directed paths.*

We now proceed by formally describing and exploring the stochastic growth of a DAG-based ledger given $m_1, ..., m_n$ strategic miners in a step-by-step fashion. As we already mentioned, we assume that the ledger grows over a finite discrete time horizon: $t = 1, ..., T$. Each turn will consist of four phases: a *block revelation phase*, in which nature picks a miner to initialise a block, an *information update phase*, where miners update their views of the block and transaction graphs, an *action phase*, in which miners employ their strategies depending on their local information, and a *natural transaction generation phase*, in which non-miners stochastically publish transactions to the P2P network.

At the end of the action phase of each turn t, we maintain a global block-DAG and transaction directed graph, denoted by G_t^{glob} and T_t^{glob} respectively. We say that the vertices of G_t^{glob} are blocks and we have that G_t^{glob} contains every block (public or private) that has been created up to turn t. Similarly, for T_t^{glob} we have that it consists of every transaction present in the network up to point t. We denote $V(G_t^{glob}) = \{B_1, ..., B_t\}$, where the i-th block was created at the i-th turn and $V(T_t^{glob}) = T_t^* \cup \overline{T_t}$, where $T_t^* = \{tx_1^*, ..., tx_t^*\}$ (enumerated) represents the set of the respective block rewards and $\overline{T_t}$ the set of the transactions.

Each block B_t, has out-degree of at most k and carries at most $\eta + 1$ transactions denoted by $Tx(B_t) \subseteq V(T_{t-1}^{glob})$ such that $tx_t^* \in Tx(B_t)$. On the other hand, the out degree of every transaction in T_t^* is 0 and the out degree of every vertex in $\overline{T_t}$ is at least 1. The reason for the aforementioned constraints on the vertices of G^{glob} and T_t^{glob} is that when a block is found, block reward is created "out of thin air", and can hence be a designated as a transaction with no dependencies on which future transactions can depend. In addition, if $A \subseteq G_t^{glob}$, we let $Tx(A) = \cup_{B_j \in V(A)} Tx(B_j)$ be the set of all induced transactions from the subgraph A. Finally, these time-evolving graphs will have the property that if $t_1 < t_2$, then $G_{t_1}^{glob} \subseteq G_{t_2}^{glob}$ and $T_{t_1}^{glob} \subseteq T_{t_2}^{glob}$.

Let us now explore both the block and the transaction directed graphs from the perspective of a miner. We suppose that each miner m_i has the following information at the end of turn t:

- $G_{i,t}^{pub}$: The DAG consisting of all blocks m_i has inferred from G_t^{glob} via the P2P network.
- $PB_{i,t} \subseteq V(G_t^{glob})$: A set of private blocks m_i has not yet shared to the P2P network.
- $T_{i,t}^{pub}$: The directed graph consisting of all transactions m_i has inferred from T_t^{glob} via the P2P network.
- $PT_{i,t} \subseteq V(T_t^{glob})$: A set of private transactions m_i has not yet shared to the P2P network.

Finally, we let G_t^{pub} and T_t^{pub} be the set of all blocks and transactions that have been shared to the P2P network respectively.

Definition 2 (Local Information). *For a given miner m_i, we let $L_{i,t} = (G_{i,t}^{pub}, PB_{i,t}, T_{i,t}^{pub}, PT_{i,t})$ and say this is the local information available to miner at the end of round t. We also say that $L_t = (L_{i,t})_{i=1}^t$ is the local information of all miners at the end of round t.*

We conclude by defining what we mean by a single-step P2P information update for a miner, as well as what the strategy space available to a miner is.

Definition 3 (Information Update). *Suppose that $H \subseteq G$ are graphs. Furthermore, suppose that the vertex set $A \in V(G) \setminus V(H)$. We define the distribution $U((H,G),A,q)$ as a single P2P information update via a specific sampling procedure. To sample $G' \sim U((H,G),A,q)$ we do the following:*

- *Let $X = \emptyset$*
- *Independently, for each $v \in A$, with probability q, add v to X.*
- *Let $G' = \Gamma(V(H) \cup X \mid G)$.*

Definition 4 (Memoryless Miner Strategies). *A miner strategy for m_j is denoted by $S_j = (S_j^I, S_j^P, S_j^T)$ and consists of an initialisation strategy S_j^I, a publishing strategy S_j^P, and a transaction creation strategy S_j^T. Each of these functions takes as input $L_{j,t} = (G_{j,t}^{pub}, PB_{j,t}, T_{j,t}^{pub}, PT_{j,t})$ at any given round t.*

- Initialisation strategy: $S_j^I(L_{j,t}) = (X^I, Y^I)$ *where* $X^I \subseteq V(T_{j,t-1}^{pub}) \cup PT_{j,t-1}$ *and* $Y^I \subseteq V(G_{j,t-1}^{pub}) \cup PB_{j,t-1}$. *Furthermore,* $|X^I| \leq \eta$ *and* $1 \leq |Y^I| \leq k$.
- Publishing strategy: $S_j^P(L_{j,t}) = (X^P, Y^P)$ *where* $X^P \subseteq PB_{j,t}$ *and* $Y^P \subseteq PT_{j,t}$. *with the property that if* $B_i \in X^P \Rightarrow tx_i^* \in Y^P$.
- Transaction creation strategy: $S_j^T(L_{j,t}) = \big(\{x_1, ..., x_k\}, \{\Gamma^1(x_1), ..., \Gamma^1(x_k)\}, W\big)$, *where each* $x_i \notin V(T_{t-1}^{priv})$, *each set* $\Gamma^1(x_i) \subseteq V(T_{t-1}^{priv})$ *is non-empty, and* $W \subseteq \{x_1, ..., x_k\}$.

To make sense of Definition 4, it suffices to note that S_j^I is invoked when m_j is chosen to mine a block. Set X^I represents the set of the transactions that the block will contain. The number of these transactions can be at most η and each block forcibly contains tx_t^*. On the other hand, set Y^I describes the set of the blocks that the newly created block will point to. The number of these blocks can be at least 1 and at most k. Moving to S_j^P, this is invoked when m_j wishes to publish hidden blocks/transactions to the P2P network. Finally, S_j^T is invoked when m_j wishes to create an arbitrary (finite) amount of new transactions $x_1, .., x_k$ that depend on transactions in $T_{j,t-1}^{priv}$ (each x_i has a non-empty set $\Gamma^1(x_i)$ of dependencies). Notice that since $\Gamma^1(x_i) \neq \emptyset$, that forcibly each x_i can not be of the form tx_r^* for some r. Finally, $W \subseteq \{x_1, ..., x_k\}$ represents which of the newly created transactions will be broadcast to the P2P network.

Due to space constraints, we refer the reader for a detailed description and a formal definition of the *ledger growth* in DAG-based ledgers to Appendix A.

3 $\mathcal{P}_{f,k}$ Ledger Models and Honest Behaviour

The main purpose of this section is twofold: first we introduce a family of honest strategies that generalise honest mining in Bitcoin and SPECTRE called $\mathcal{P}_{f,k}$ mining, and second we introduce constraints on $\mathcal{D}$ that represent honest transaction generation by end-use agents in a DAG-based ledger (this includes Bitcoin and SPECTRE as well).

Definition 5 (Depth and Weight of a Block). *Suppose that $G \in \mathcal{G}$ is a block-DAG. In other words, G is connected and has a genesis block B_0. For a given $B_t \in G$, we let $w(B_t) = |\mathit{\Gamma}(\{B_t\} \mid G))| - 1$ be the weight of B_t. This is the number of predecessors B_t has in G. We also define $D(B_t) = d_G(B_t, B_0)$ as the depth of B_t. This is the graphical distance between B_t and B_0, i.e. the length of the (unique) shortest path between B_t and B_0 in G.*

In Bitcoin, miners resolve ambiguity in ledger consensus by initialising found blocks to point to the longest chain in the DAG. One reason for this is that agents have provably used significant computational power to grow said chain, and re-writing this history is thus computationally infeasible. In DAG-based ledgers, agents may point to multiple blocks. Thus, following this same thought process, they should point to blocks with a provably significant amount of computation in their histories. The issue, however, is that measuring how much computation exists in the past of a leaf is ambiguous in DAGs: a block could have either large weight or large depth (unlike in Bitcoin where these quantities are always the same), and it is unclear to decide which to give precedence to. In order to completely rank the importance of leaves in a block DAG, we simply use a family of score functions that expresses convex combinations of depth and weight.

Definition 6 (Score Function). *Suppose that $\alpha \in [0, 1]$ and $\beta = 1 - \alpha$. We say that f is an (α, β) block-DAG score function if for a given block-DAG, $G \in \mathcal{G}$, $f(B_t) = \alpha D(B_t) + \beta w(B_t)$.*

In a nutshell, honest block-DAG growth in $\mathcal{P}_{f,k}$ protocols with parameter α and β prescribes that miners prepare blocks with at most k pointers that point the locally visible blocks in the block-DAG that with highest score under f.

3.1 Valid Blocks and Transactions

In ledgers employing decentralised consensus protocols, there is an explicit consensus mechanism whereby agents are able to look at their local view of the ledger and extrapolate valid blocks and subsequently valid transactions within the local view of the ledger. In Bitcoin for example, valid blocks consist of the longest chain in the ledger, and valid transactions consist of transactions within said longest chain. SPECTRE, on the other hand, has any seen block as valid, but the valid transaction extraction process is a complicated voting procedure that extracts a subset of transactions within the local view of the DAG as valid. We proceed by providing a definition of valid block and transaction extractors in $\mathcal{P}_{f,k}$ models that generalises both of these examples.

Definition 7 (Valid Block Extractors and Valid Transaction Extractors). *Suppose that G is a block-DAG and that $\ell_1, .., \ell_k$ are the k leaves in G that have the highest score under f. Then we say that $VB(G) = \mathit{\Gamma}(\{\ell_1, ..., \ell_k\} \mid G)$ is the DAG of valid blocks in G under $\mathcal{P}_{f,k}$. In addition, we let $VT(G) \subseteq Tx(VB(G))$ be the set of valid transactions for a specified transaction extractor function VT. We say that VT is in addition monotonic if it holds that if $VB(H) \subseteq VB(G)$, then $VT(H) \subseteq VT(G)$.*

In what follows we define a special type of monotonic valid transaction selection rule called *present transaction selection*. The reason we outline this simple selection rule is that in Section 3.3 we will show that if all miners employ monotonic valid transaction selection and the honest strategies presented in Section 3.2, then we can assume without loss of generality that they employ present transaction selection as a valid transaction selection rule.

Definition 8 (Present Transaction Selection). *Suppose that G is a block-DAG and that $\ell_1, .., \ell_k$ are the leaves in G that have the highest score under f. Then we say that $VB(G) = \Gamma(\{\ell_1, ..., \ell_k\} \mid G)$ is the DAG of valid blocks in G under $\mathcal{P}_{f,k}$. In addition we say that $PVT(G) = Tx(VB(G))$ is the set of present valid transactions in G under $\mathcal{P}_{f,k}$.*

3.2 Defining S^I, S^P, and S^T for Honest Mining in $\mathcal{P}_{f,k}$

We define $H_j = (H_j^I, H_j^P, H_j^T)$ as the honest strategy employed by m_j in $\mathcal{P}_{f,k}$, and describe each component below.

- H_j^I: Compute $A = VB(G_{j,t}^{pub})$ and $B = VT(G_{j,t}^{pub})$. Let $H_j^I(L_{j,t}) = (X, Y)$. X is the set of at most η oldest non-block-reward (i.e. not of the form tx_r^*) transactions in $T_{j,t}^{pub} \setminus B$ (ties are broken arbitrarily) with a graphical closure in B. $Y = \{\ell_1, ...\ell_k\}$ is the set of k highest-score leaves in $G_{j,t}^{pub}$ under f.
- H^P: Publish all private blocks and transactions immediately
- H^T: Create no new transactions (the assumption is that transactions created by pools are negligible with respect to the total transaction load of the ledger)

Before continuing, we note that in the $\mathcal{P}_{f,k}$ model, H^T ensures that honest miners do not create and broadcast any transactions themselves. This, of course, is not the case in practice, but it is an accurate approximation to a regime in which the fraction of transactions created by miners is a negligible fraction of all transactions created by end-users of the ledger.

Also notice that H_j^I dictates that the oldest transactions will be included to agents' j block. We make this choice for simplicity reasons, however we want to point out that a more sophisticated selection strategy may be more beneficial for the protocol, especially in terms of *efficiency* (as we will see in Section 4).

Implementation of Bitcoin and SPECTRE as $\mathcal{P}_{f,k}$ Protocols. With the previous machinery in place, we can see that block-DAG and transaction-DAG growth in Bitcoin and SPECTRE are special cases of $\mathcal{P}_{f,k}$ ledgers. For Bitcoin, we let $k = 1$, and any parameter setting, (α, β) for f results in Bitcoin growth. As for SPECTRE, we let $k = \infty$ and once more any parameter setting (α, β) for f suffices to implement honest SPECTRE ledger growth.

3.3 Honest Transaction Consistency and Generation

As mentioned in Section 3.1, we can show that amongst monotonic transaction extractors, present transaction extractors are all we need for honest ledger growth in the $\mathcal{P}_{f,k}$ model.

Theorem 1. *If the valid transaction extractor, VT, is monotonic and all miners employ $H = (H^I, H^P, H^T)$, then VT is a present transaction extractor.*

Proof. Suppose that $L_{i,t} = (G_{i,t}^{pub}, PB_{i,t}, T_{i,t}^{pub}, PT_{i,t})$ is the local information available to m_i at turn t. Since m_i is honest, one can easily see that $G_{i,t}^{pub} = G_{i,t}^{priv} = G_{i,t}$ and $T_{i,t}^{pub} = T_{i,t}^{priv} = T_{i,t}$. Clearly $VT(G_{i,t}) \subseteq PVT(G_{i,t})$. Now suppose that $x \in PVT(G_{i,t})$. This means that $x \in Tx(B_r)$ for some block B_r found by say m_j. This means that in turn r, m_j invoked H^I to create B_r, which means that since $x \in Tx(B_r)$, all dependencies of x are in $VT(\Gamma(B_r \mid G_{i,t}))$, the valid transactions from the DAG consisting of the closure of B_r in the block DAG. However $VT(\Gamma(B_r \mid G_{i,t})) \subseteq VT(G_{i,t})$ since VT is monotonic. Therefore x has its dependencies met in $VT(G_{i,t})$ so that $x \in VT(G_{i,t})$. This implies $VT(G_{i,t}) = PVT(G_{i,t})$ as desired. □

In light of this theorem, we focus on monotonic valid transaction extractors given their generality. Hence, from now on we assume that when we invoke VT, we in fact mean that VT is a present transaction extractor.

Regarding the honest transaction generation, H^T dictates that each m_i does not produce or propagate transactions created by themselves. Hence, it is crucial that we properly define $\mathcal{D}$ in the $\mathcal{P}_{f,k}$ model. At first one may be tempted to simply treat the random growth of T_t^{glob} as independent of G_t^{glob}, but this is a grave mistake. To see why, imagine that G_t^{glob} contains some block B_r that is orphaned by each m_i (note that this can only happen if $k < \infty$). If the growth of T_t^{glob} is independent of that of G_t^{glob}, then it could be the case that many (if not infinitely many) future transactions depend on t_r^*. However, if B_r is orphaned by all miners, tx_r^* is not valid, hence none of these future transactions will be added to the ledger via close inspection of how H^I is defined.

A compelling fact is that if all miners have orphaned B_r, then chances are that whatever local view of G_t^{pub} an end-user has, they too will have orphaned B_r, and thus will not have tx_r^* as a valid transaction. In more direct terms, any money created via the block reward of B_r is not actually in the system for an end-user, so if this end-user is honest, there is no reason why they would produce transactions that would depend on this illegitimate source of currency.

Definition 9 (Honest Transaction Distributions). *Let G_t^{glob} be a global block DAG at turn t with k highest leaves are $\ell_1, ..., \ell_k$. In an honest setting, $VB(G_t^{glob}) = \Gamma(\{\ell_1, ..., \ell_l\} \mid G_t^{glob})$ and $VT(G_t^{glob}) = Tx(VB(G_t^{glob}))$. We say that $\mathcal{D}(G_t^{glob}, T_t^{glob})$ is an honest transaction distribution if $x \sim \mathcal{D}(G_t^{glob}, T_t^{glob})$ is such that $x \notin T_t^{glob}$ and its dependencies lie strictly in $VT(G_t^{glob})$.*

3.4 Assumptions: Non-Atomic Miners, Payoffs and Transaction Generation Rate

Non-Atomic Miners. For our simulations we assume that a set of honest miners, each of whom has small enough hash power, can be modelled as one larger miner who re-samples their view of both DAGs each time they are chosen for a block initialisation. This is reasonable if, for example, each miner in said collection finds at most one block in time horizon $t = 1, ..., T$ with high probability. We call these miners *non-atomic*. Due to space constraints, we refer the reader to Appendix B for a more detailed description.

Block Rewards and Transaction Fees. We suppose that at time-step T, miners get a normalised block reward of 1 per block that they have in $VB(G_T^{glob})$. As for transaction fees, the full generality of $P_{f,k}$ protocols only specifies how to extrapolate valid transactions conditional upon everyone being honest, and not who receives transaction fees (this is subsumed in the details of VT in the general setting). For this reason we further assume that transaction fees are negligible in comparison to block rewards over the time horizon $t = 1, ..., T$.

Transaction Generation Rate. Although in full generality there is no restriction on how many transactions nature may create in a given turn, we impose a fixed constraint on this quantity: λ. As such, each turn introduces $\{x_{t,1}, ..., x_{t,\lambda}\}$ transactions sampled from a specified honest transaction distribution $\mathcal{D}$. Furthermore, in our simulations we let $\lambda = \eta$, so that the ledger infrastructure can, in theory, cope with the transaction load if all miners have full information, and thus we can see specifically it falls short of this objective in the partial information setting.

4 Results

Due to lack of space, we refer the reader to Appendix C for the implementation details and assumptions we make in simulating $\mathcal{P}_{f,k}$ ledgers.

Fairness. We recall that one of the key properties of Bitcoin is that it is fair: miners earn block reward proportional to the computational resources they expend on extending the ledger. One of the most significant observations from our simulations is that $\mathcal{P}_{f,k}$ ledgers are not necessarily fair as soon as the agents begin having informational parameters, $q < 1$, as is the case in a high throughput setting. To illustrate this phenomenon, we study a two-miner scenario with agents m_0 and m_1 of hash power $(1 - h_1, h_1)$ and informational parameters (q_0, q_1). m_0 is modelled as a non-atomic miner and we empirically compute the surplus average block reward of m_1 relative to the baseline h_1 they would receive in a fair protocol. Our results are visualised in Figure 1. Each row of the figure represents $k = 1, 2, 3$ respectively and each column represents $q_0 = 0.005, 0.05, 0.2$. Each individual heatmap fixes k and q_0 and plots average block reward surplus for m_1 as $q_1 \in [0, 1]$ and $h_1 \in (0, 0.5]$ are allowed to vary. Finally, each pixel contains the average block reward surplus for $T = 50$ and averaged over 50 trials. We notice that an added strength to our fairness result is that they hold, irrespective of the underlying honest transaction distribution $\mathcal{D}$ used in practice.

The most jarring observation is that, depending on the parameters, m_1 earns a vastly different average block reward than their fair share h_1. In fact, for fixed k and q_0, there seem to be three regions of the hash space $h_1 \in (0, 0.5]$ with qualitatively distinct properties:

– If h_1 is large enough, m_1 strictly benefits from having lower q_1 values. This is due to the fact that an honest miner with small q_0 necessarily sees his own blocks and is inadvertently acting somewhat "selfishly". Hence if their hash rate is high enough, their persistent mining upon their own blocks may end up orphaning other blocks and give them a higher share of valid blocks in the final DAG.
– If h_1 is small enough, m_1 strictly benefits from having higher q_1 values. Contrary to the previous point, at small hash values, m_1 only finds a few blocks, and hence they risk losing their entire share of blocks if these blocks aren't well positioned in the block DAG, since they are in no position to inadvertently overtake the entire DAG via pseudo-selfish behaviour resulting from low q_1 values.
– Finally, for intermediate h_1 values, m_1 no longer has a monotonic surplus with respect to q_1 but rather a concave dependency. This can be seen as an interpolation of the previous two points.

We notice that where these qualitative regions of h_1 values lie within $(0, 0.5]$ depends entirely on k and q_0. In general, for fixed k (i.e specific rows within Figure 1), as q_0 increases, the transitions between these regions shift rightwards, and for fixed q_0 (i.e. specific columns in Figure 1) as k increases, also shifts rightwards, as increasing k can be seen to informally have the same effect as uniformly increasing q_0 and q_1 as agents are more likely to see blocks due to multiple pointers. Of course, for $k = \infty$ the protocol becomes fair, as every block eventually joins the DAG. As a final observation, roughly speaking, "small" miners benefit from increasing their connectivity to the P2P network, rather than investing in extra hash power, while for large miners it is the opposite.

Remark 1. In practice, the parameter k would depend on the q_i's as well as many other aspects of the network. We present a variety of results, with the throughput ranging from relatively tame to pushed beyond what the network can handle, where rampant strategic mining becomes an issue more important than fairness. This is by design: assuming honest behaviour and fixing k for different q_i's allows us to measure the worst-case improvement in fairness, even for cases that would rarely appear in reality. Moreover, our results only cover *block reward* fairness. When transaction fee rewards are included the resulting setting is far more complicated, as a limited view of the network means that even though no blocks are orphaned (for large enough k) there is no guarantee transaction rewards are fairly distributed.

Efficiency. DAG-based ledgers have been created with the aim of tackling a higher transaction load in cryptocurrencies. Given that we have a way of modelling honest transaction growth, there are three different metrics we use to precisely quantify how well DAG-based ledgers deal with a higher throughput of transactions. The first and most important is the *Proof of Work Efficiency.* More specifically, for a given DAG-based Ledger, we say that the PoW efficiency is the fraction of globally valid transactions that are present within the valid sub DAG of the block DAG, over all published transactions.

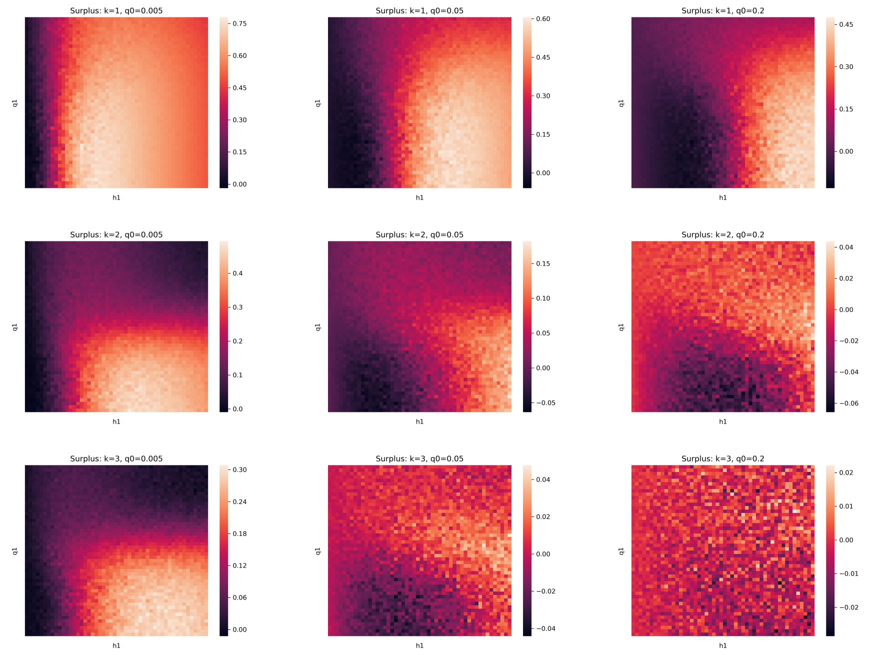

Fig. 1. Surpluses for $k = 1, 2, 3$ at $q_0 \in \{0.005, 0.05, 0.2\}$

This is the most important metric, since the goal of a ledger is to maximise the rate at which new transactions are processed. We also compare ledgers in terms of the average fraction of orphaned blocks they create and their transaction *lag*, which is defined as the time difference between the issue and successful inclusion of the DAG's most recent transaction and the final turn of the time horizon.

For our experiments, we compared $\mathcal{P}_{f,k}$ performance for $k \in \{1, 2, 3, \infty\}$ and $n = 4$ atomic miners each with $h_i = 1/4$ and varying q_i's. For all graphs, we have $\eta = 6, T = 100$ and the results have been averaged over 50 trials. First of all,

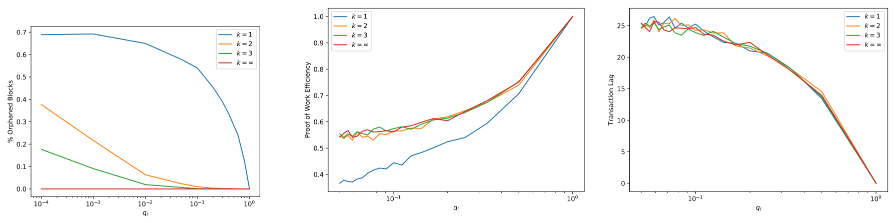

Fig. 2. Performance Metrics for $n = 4$ miners and $k \in \{1, 2, 3, \infty\}$

we notice that for all parameter settings of $\mathcal{P}_{f,k}$, there exist information regimes where if each q_i is low enough, the ledger suffers in its efficiency–even in the case where $k = \infty$. We also observe that increasing k improves all metrics except lag, but not dramatically. For reasonable values of q_i, before fairness becomes an issue, there is a significant performance increase between $k = 1$ and $k \geq 2$. However, $k > 2$ is only really necessary for extremely small q_i.

Remark 2. This result depends on the assumption that miners add the *oldest* transactions they can, every time they mine a block. A more sophisticated strategy, such as greedily adding the more valuable transactions or a mix from different branches (as in [16]) could improve efficiency for higher k.

We also compared the performance for $n \in \{1, 2, \ldots, 20\}$ with $q = h = 1/n$, leading to similar results. Notably, as the number of miners grows the number of orphaned blocks decreases and the PoW efficiency improves with k.

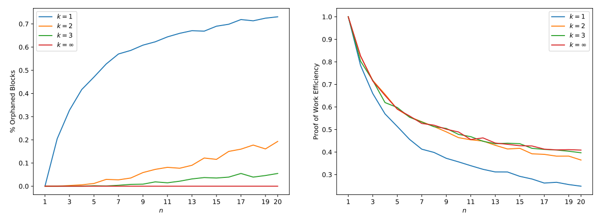

Fig. 3. Performance Metrics for $n \in \{1, 2, \ldots, 20\}$ and $k \in \{1, 2, 3, \infty\}$

Dynamically adjusting k and f. A key feature of Bitcoin is its dynamically adjusted difficulty. Our results suggest that a DAG-based ledger may also be able to dynamically adjust its internal parameters k and f to cope with changing transaction loads from end users. Even though larger values of k do not always provide a significant advantage, a dynamically adjusted protocol could sacrifice block size to make room for more pointers if efficiency is suffering in a period of high transaction loads to the ledger.

5 Discussion

Given our results, it would be interesting to modify $\mathcal{P}_{f,k}$ ledgers and quantify the improvements to efficiency and fairness. For example, if we employ more sophisticated strategies for transaction inclusion, what is the behaviour of the ledger in terms of efficiency? In addition, what happens if we augment the ledger space with a different and more complicated class of score functions? By conditioning k to the informational parameters q_i, capturing the essence of existing DAG protocols more concretely, how fairness is affected in case we no longer assume negligible transaction fees? It would also be interesting to drop the honest mining assumption and explore this model in full generality.

Bibliography

[1] Moshe Babaioff, Shahar Dobzinski, Sigal Oren, and Aviv Zohar. On bitcoin and red balloons. In *Proceedings of the 13th ACM Conference on Electronic Commerce, EC 2012, Valencia, Spain, June 4-8, 2012*, pages 56–73, 2012.

[2] Georgios Birmpas, Elias Koutsoupias, Philip Lazos, and Francisco J Marmolejo-Cossío. Fairness and efficiency in dag-based cryptocurrencies. *arXiv preprint arXiv:1910.02059*, 2019.

[3] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 104–121, 2015.

[4] Miles Carlsten, Harry Kalodner, S Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 154–167. ACM, 2016.

[5] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *13th IEEE International Conference on Peer-to-Peer Computing, IEEE P2P 2013, Trento, Italy, September 9-11, 2013, Proceedings*, pages 1–10, 2013.

[6] ethereum/wiki. A next-generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/WhitePaper/., October 2015.

[7] Ittay Eyal. The miner's dilemma. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 89–103, 2015.

[8] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*, pages 436–454, 2014.

[9] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, Santa Clara, CA, USA, March 16-18, 2016*, pages 45–59, 2016.

[10] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 281–310, 2015.

[11] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 51–68, 2017.

[12] Kolbeinn Karlsson, Weitao Jiang, Stephen B. Wicker, Danny Adams, Edwin Ma, Robbert van Renesse, and Hakim Weatherspoon. Vegvisir: A partition-tolerant blockchain for the internet-of-things. In *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018*, pages 1150–1158, 2018.

[13] Aggelos Kiayias and Giorgos Panagiotakos. On trees, chains and fast transactions in the blockchain. In *Progress in Cryptology - LATINCRYPT 2017 - 5th International Conference on Cryptology and Information Security in Latin America, Havana, Cuba, September 20-22, 2017, Revised Selected Papers*, pages 327–351, 2017.

[14] Aggelos Kiayias, Elias Koutsoupias, Maria Kyropoulou, and Yiannis Tselekounis. Blockchain mining games. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 365–382. ACM, 2016.

[15] Elias Koutsoupias, Philip Lazos, Foluso Ogunlana, and Paolo Serafino. Blockchain mining games with pay forward. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 917–927.

[16] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, pages 528–547, 2015.

[17] Hanqing Liu, Na Ruan, Rongtian Du, and Weijia Jia. On the strategy and behavior of bitcoin mining with n-attackers. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 357–368. ACM, 2018.

[18] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[19] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 305–320, 2016.

[20] Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, pages 39:1–39:16, 2017.

[21] Yonatan Sompolinsky and Aviv Zohar. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. Cryptology ePrint Archive, Report 2013/881, 2013. https://eprint.iacr.org/2013/881.

[22] Yonatan Sompolinsky and Aviv Zohar. Phantom: A scalable blockdag protocol. Cryptology ePrint Archive, Report 2018/104, 2018. https://eprint.iacr.org/2018/104.

[23] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre: A fast and scalable cryptocurrency protocol. Cryptology ePrint Archive, Report 2016/1159, 2016. https://eprint.iacr.org/2016/1159.

[24] Florian Tschorsch and Björn Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys and Tutorials*, 18(3):2084–2123, 2016.

A Ledger Growth per State

In order to describe and explain ledger growth in DAG-based ledgers in more detail, we introduce the notion of a ledger state and describe how the ledger transitions from state to state. More specifically, at the end of round t, we say the ledger is in state $\sigma_t = \{L_t, G_t^{glob}, T_t^{glob}, G_t^{pub}, T_t^{pub}\}$. Recall that as we already mentioned in Section 2.1, each turn t consists of four different phases.

The Genesis Block and the Beginning of the Ledger: $\boldsymbol{\sigma_0}$ We bootstrap the ledger by creating a genesis block, B_0, which comes along with a genesis transaction tx_0^*. As such, $V(G_0^{pub}) = B_0$, $E(G_0^{pub}) = \emptyset$, $V(T_0^{pub}) = tx_0^*$, and $E(T_0^{pub}) = \emptyset$. In addition, $L_{0,i} = (G_0^{pub}, \emptyset, tx_0^*, \emptyset)$ for each m_i, $G_0^{pub} = G_0^{glob}$, and $T_0^{pub} = T_0^{glob}$, thus fully defining σ_0.

The Mining Phase of Round $\boldsymbol{t}$: $\boldsymbol{\Delta_1}$ Suppose that σ_{t-1} is given. First a random miner is chosen, where the probability that each m_i is chosen is precisely h_i. Suppose m_j is drawn, the block B_t is initialised as follows: $Tx(B_t) = \pi_1(S_j^I(L_{j,t})) \cup tx_t^*$ and we update the global block DAG by letting $V(G_t^{glob}) = V(G_{t-1}^{glob}) \cup B_t$ and $E(G_t^{glob}) = E(G_{t-1}^{glob}) \cup \{(B_t, y) \mid y \in \pi_2(S_j^I(L_{j,t}))\}$. In addition, for $r \neq j$, we let $L'_{r,t-1} = L_{r,t-1}$, and for j, we let $L'_{j,t-1} = ((G_{j,t-1}^{pub}, PB_{j,t-1} \cup B_t), (T_{j,t-1}^{pub}, PT_{j,t-1} \cup tx_t^*))$, so that $L'_{t-1} = (L'_{i,t-1})_{i=1}^n$. With this in hand, we can express the randomised transition $\Delta_1(\sigma_{t-1})$ as follows: $\Delta_1(\sigma_{t-1}) = \{L'_{t-1}, G_t^{glob}, T_{t-1}^{glob}, G_{t-1}^{pub}, T_{t-1}^{pub}\}$

The Information Update Phase of Round $\boldsymbol{t}$: $\boldsymbol{\Delta_2}$ Suppose that $\Delta_1(\sigma_{t-1}) = \{L'_{t-1}, G_t^{glob}, T_{t-1}^{glob}, G_{t-1}^{pub}, T_{t-1}^{pub}\}$ is given after block initialisation. For each miner m_i, we sample $A_i \sim U\left((G_{i,t-1}^{pub}, G_{t-1}^{pub}), V(G_{t-1}^{pub}) \setminus V(G_{i,t-1}^{pub}), q_i\right)$ from the block-DAG, and we sample $C_i \sim U\left((T_{i,t-1}^{pub}, T_{t-1}^{pub}), V(T_{t-1}^{pub}) \setminus V(T_{i,t-1}^{pub}), q_i\right)$ from the transaction DAG. We also let $F_i = \Gamma(V(C_i) \cup Tx(A_i) \mid T_{t-1}^{glob})$. Accordingly, we can define $L_{i,t}'' = ((A_i, PB_{i,t-1}), (F_i, PT_{i,t-1}))$, and $L_{i,t-1}'' = (L_{t-1}'')_{i=1}^n$. With this in hand, we can express the randomised transition $\Delta_2 \circ \Delta_1(\sigma_{t-1})$ as follows: $\Delta_2 \circ \Delta_1(\sigma_{t-1}) = \{L_{t-1}'', G_t^{glob}, T_{t-1}^{glob}, G_{t-1}^{pub}, T_{t-1}^{pub}\}$

The Action Phase of Round $\boldsymbol{t}$: $\boldsymbol{\Delta_3}$ Let $\Delta_2 \circ \Delta_1(\sigma_{t-1}) = \{L_{i,t}'', G_t^{glob}, T_{t-1}^{glob}, G_{t-1}^{pub}, T_{t-1}^{pub}\}$ be given after an information update phase. We first update the global transaction graph by letting the vertices of a new graph be:

$$V(T_t^{glob'}) = V(T_{t-1}^{glob}) \cup \{tx_t^*\} \cup \left(\cup_{i=1}^n \pi_2(S_i^P(L_{i,t-1}''))\right) \cup \left(\cup_{i=1}^n \pi_3(S_i^T(L_{i,t-1}''))\right)$$

and letting the edges of the new graph be:

$$E(T_t^{glob'}) = E(T_{t-1}^{glob}) \cup \left(\cup_{i=1}^n E(\pi_2(S_i^P(L_{i,t-1}'')))\right) \cup \left(\cup_{i=1}^n E(\pi_3(S_i^T(L_{i,t-1}'')))\right)$$

where we slightly abuse notation and for a set of transactions, A, we denote $E(A)$ as the set of all tuples (x, v) such that $x \in A$ and v is a transaction x depends on. With this in hand, We can define the following objects according to the deterministic strategies S_i of each m_i:

- $G^{pub}_{i,t} = \Gamma(V(G^{pub}_{i,t-1}) \cup \pi_1(S^P_i(L_{i,t-1}'')) \mid G^{glob}_t)$
- $PB_{i,t} = PB_{i,t-1}'' \setminus \pi_1(S^P_i(L_{i-,t-1}''))$
- $T^{pub}_{i,t} = \Gamma(V(T^{pub}_{i,t-1}) \cup \pi_2(S^P_i(L_{i,t-1}'')) \mid T^{glob'}_t)$
- $PT_{i,t} = PT_{i,t-1} \setminus \pi_2(S^P_i(L_{i-,t-1}''))$

Finally, we let $L_{i,t} = ((G^{pub}_{i,t}, PB_{i,t}), (T^{glob}_{i,t}, PT_{i,t}))$, $L_t = (L_{i,t})^n_{i=1}$, $G^{pub}_t = \cup^n_{i=1} G^{pub}_{i,t}$ and $T^{pub'}_t = T^{pub}_{t-1} \cup \left(\cup^n_{i=1} T^{pub}_{i,t}\right)$. With this in hand, we can express the transition $\Delta_3 \circ \Delta_2 \circ \Delta_1(\sigma_{t-1})$:

$$\Delta_3 \circ \Delta_2 \circ \Delta_1(\sigma_{t-1}) = \{L_t, G^{glob}_t, T^{glob'}_t, G^{pub}_t, T^{pub'}_t\}$$

Nature Adds Transactions to $\boldsymbol{T^{glob}_t}$: $\boldsymbol{\Delta_4}$ Finally, we model transactions that are naturally produced by agents other than miners, $m_1, ..., m_n$ (the end-users of the ledger) and broadcast via the P2P network. To this end, we suppose that $\mathcal{D}$ is a pre-defined function, such that $\mathcal{D}(G^{glob}_t, T^{glob'}_t)$ is a distribution that returns $(\{x_1, ..., x_k\}, \{\Gamma^1(x_1), ..., \Gamma^1(x_k)\})$, a random finite set of new transactions $\{x_1, ..., x_k\}$ along with their dependencies $\{\Gamma^1(x_1), ..., \Gamma^1(x_k)\}$, with the property that $\Gamma^1(x_i) \neq \emptyset$ for each x_i. This latter point ensures that no x_i is of the form tx^*_r. With this in hand, we let $A \sim \mathcal{D}(G^{glob}_t, T^{glob'}_t)$ and define $V(T^{glob}_t) = T^{glob'}_t \cup \pi_1(A)$, $E(T^{glob}_t) = E(T^{glob'}_t) \cup \pi_2(A)$, $V(T^{pub}_t) = T^{pub'}_t \cup \pi_1(A)$, and $E(T^{pub}_t) = E(T^{pub'}_t) \cup \pi_2(A)$. Ultimately, this allows us to define

$$\Delta_4\left(\{L_t, G^{glob}_t, T^{glob'}_t, G^{pub}_t, T^{pub'}_t\}\right) = \{L_t, G^{glob}_t, T^{glob}_t, G^{pub}_t, T^{pub}_t\}$$

Putting Everything Together. In the previous sections we have detailed every process of the evolution of a DAG-based ledger. Before stating the formal definition in its full generality, we recall that the aforementioned growth process has several parameters: $\theta = (\boldsymbol{m}, \boldsymbol{h}, \boldsymbol{q}, \boldsymbol{S}, \eta, k, \mathcal{D})$.

- $\boldsymbol{m} = (m_1, ..., m_n)$ are strategic miners.
- $\boldsymbol{h} = (h_1, ..., h_n)$ are the hash rates of each miner.
- $\boldsymbol{q} = (q_1, ..., q_n)$ is the informational parameter of each miner.
- $\boldsymbol{S} = (S_1, ..., S_n)$ are ledger-building strategies for strategic miner
- $\eta \geq 1$ is the maximum number of transactions in a single block.
- $k \geq 1$ is the maximum out-degree of a block in the block DAG.
- $\mathcal{D}$ is a natural transaction generation distribution as per above.

To avoid confusion, we let $\Delta_\theta = \Delta_4 \circ \Delta_3 \circ \Delta_2 \circ \Delta_1$, with the influence of θ implicit in our previous exposition. With this we are able to fully formulate ledger growth in its complete generality.

Definition 10 (Formal Ledger Growth). *Suppose $\theta = (\boldsymbol{m}, \boldsymbol{h}, \boldsymbol{q}, \boldsymbol{S}, \eta, k, \mathcal{D})$. We let $\mathcal{P}_\theta$ denote DAG-based ledger growth governed by θ and define it recursively as follows: We let σ_0 be defined as per Section A, and consecutively let $\sigma_t = \Delta^t_\theta(\sigma_0)$ for $t \geq 1$.*

B Non-Atomic Miners

It is common in the analysis of Bitcoin and related cryptocurrencies to group an arbitrary number of honest miners into one honest miner with the aggregate hash power of all different honest miners. The behaviour of multiple honest miners or one aggregate honest miner is indistinguishable from the local perspective of a single strategic miner. In the DAG-based ledger one could perform a similar analysis, however the partial information inherent in our model does not allow us to directly do so. The reason for this is that separate honest miners may have different partial views of the block DAG and transaction DAG. However, if for a given time horizon $t = 1, ..., T$, it is the case that a group of honest miners each have a small enough hash power that they most likely never see more than at most one block in the time horizon, we can aggregate all said miners into one large honest miner that simply re-samples their view of the block DAG and transaction DAG whenever they are chosen to initialise a block and invoke H^I. We call such miners non-atomic and formalise their definition below.

Definition 11 (Non-atomic miner). *For a given time-horizon, $t = 1, ..., T$, we say that a group of miners $m'_1, ..., m'_k$ is non-atomic if with high probability, each such miner finds at most one block in this time horizon.*

Simulating Non-Atomic Miners. When a group of non-atomic miners finds a block they need to create a fresh sample of what partial information they have in the block/transaction DAG. To do so, let us suppose that $m'_1, ..., m'_k$ are a group of non-atomic miners which we group together into a single miner, m^* of hash power $\sum_{i=1}^k h_i$. Furthermore, we suppose that each non-atomic miner in this group has informational parameter q. When m^* finds a block at time-step $t = 1, ..., T$, we simply take each block/transaction present in the global block/transaction DAG, and for each turn it has been present in its respective global DAG, flip a biased coin of weight q to see whether m^* directly sees this block/transaction. Once this preliminary list of seen blocks/transactions is created, m^* also sees all ancestors of said blocks/transactions.

C Computationally Modelling Honest Growth in $\boldsymbol{\mathcal{P}_{f,k}}$ Ledgers

We describe pseudo-code for honest ledger growth in $\mathcal{P}_{f,k}$ ledgers. In the implementation described in Algorithm 1 we make the following assumptions:

– The underlying ledger is $\mathcal{P}_{f,k}$

- All miners are Honest
- VT is a present transaction extractor
- The number of transactions created by miners is negligible with respect to the total number of transactions created within the P2P network.
- The honest transaction generation distribution, $\mathcal{D}(G_t^{glob}, T^{glob})t)$, is modelled as follows: for a given transaction, tx, to be sampled via $\mathcal{D}$, first $k \sim Poiss(\gamma)$ is drawn as the number of dependencies tx will have from $VT(G_t^{glob})$, and subsequently, $x_1, ..., x_k$ are drawn from $VT(G_t^{glob})$ uniformly randomly, where we allow repetitions. Ultimately we let $\{y_1, .., y_r\}$ be the unique transaction set extrapolated from $\{x_1, ..., x_k\}$ and this is precisely the set of dependencies of the newly sampled tx. We note that transactions can depend on the block reward of arbitrarily old blocks. Hence uniformity is not an unfeasible assumption in sampling dependencies of a new transaction given our time horizons. Finally, in our simulations, since $T = 50$, we let $\gamma = 2$ so that transactions on average have multiple dependencies on average.

Given these assumptions, we also condense the notation used for key variables needed for ledger growth in the pseudo-code for Algorithm 1. In particular:

- G_t and T_t are the global block and transaction DAGs respectively
- $G_{i,t}^{vis}$ and $T_{i,t}^{vis}$ are the portions of G_t and T_t visible to m_i at the end of turn t.
- $Owner(B_t) = i$ means that m_i mined block B_t

Algorithm 1 Honest Ledger Growth

```
Require:
Ledger growth Parameters: $\theta = (\boldsymbol{\alpha}, \boldsymbol{q}, \eta, k, \mathcal{D}, \lambda, T)$

Genesis:
$G_0 \leftarrow (\{B_0\},\ \emptyset),\ T_0 \leftarrow (\{tx_0^*\}, \emptyset),\ Tx(B_0) \leftarrow tx_0^*$
for $j = 1, \ldots, N$ do
    $G_{i,0}^{vis} \leftarrow G_0$
    $T_{i,0}^{vis} \leftarrow T_0$

for $t = 1, \ldots, T$ do
    Mining Phase:
    Sample $i$ from $\boldsymbol{\alpha}$
    Compute $\ell_1, ..., \ell_r$, top $r \leq k$ leaves of $G_{i,t-1}^{vis}$ in terms of score, $f$
    Compute $tx_{i_1}, ..., tx_{i_s}$, with $s \leq \eta$, oldest pending transactions of non-zero out-
    degree in $T_{i,t-1}^{vis}$ (ties broken lexicographically)
    $G_t \leftarrow (V(G_{t-1}) \cup B_t,\ E(G_{t-1}) \cup \{(B_t, \ell_1), ..., (B_t, \ell_r)\})$
    $Tx(B_t) = tx_t^* \cup \{tx_{i_1}, ..., tx_{i_s}\}$
    $Owner(B_t) = i$
    $V(G_{i,t-1}^{vis}) \leftarrow B_t,\ V(T_{i,t-1}^{vis}) \leftarrow tx_t^*$

    Tx Generation:
    Compute valid transactions: $VT_{t-1} \subseteq V(T_{t-1})$
    $T_t \leftarrow (V(T_{t-1}) \cup \{tx_t^*\},\ E(T_{t-1}))$
    for $j = 1, \ldots, \lambda$ do
        Use $\mathcal{D}$ and $VT_{t-1}$ to draw $\{tx_{i_1}, ..., tx_{i_s}\}$ dependencies of $tx_{t,j}$
        $V(T_t) \leftarrow V(T_t) \cup \{tx_{t,j}\},\ E(T_t) \leftarrow E(T_t) \cup \{(tx_{t,j}, tx_{i_1}), ..., (tx_{t,j}, tx_{i_s})\}$

    Information Update:
    for $j = 1, \ldots, N$ do
        $G_{j,t}^{vis} \leftarrow G_{j,t-1}^{vis},\ T_{j,t}^{vis} \leftarrow T_{j,t-1}^{vis}$
        for each $B_r \in V(G_t) \setminus V(G_{j,t}^{vis})$ do
            With probability $q_i$: $V(G_{j,t}^{vis}) \leftarrow V(G_{j,t}^{vis}) \cup \{B_r\}$
        for each $tx_s \in V(T_t) \setminus V(T_{j,t-1}^{vis})$ do
            With probability $q_i$: $V(T_{j,t}^{vis}) \leftarrow V(T_{j,t}^{vis}) \cup \{tx_s\}$
        $G_{j,t}^{vis} \leftarrow \mathit{\Gamma}(V(G_{j,t}^{vis}), G_t)$
        $T_{j,t}^{vis} \leftarrow \mathit{\Gamma}(Tx(G_{j,t}^{vis}) \cup V(T_{j,t}^{vis}), T_t)$

return $G_T, T_T, \{Owner(B_t)\}_{t=1}^{T}, \{Tx(B_t)\}_{t=1}^{T}, \{G_{i,T}^{vis}\}_{i=1}^{N}, \{T_{i,T}^{vis}\}_{i=1}^{N}$
```