applied system innovation

MDPI

Article

Resilience Analysis for Double Spending via Sequential Decision Optimization

Juri Hinz †

School of Mathematical and Physical Sciences, University of Technology Sydney, P.O. Box 123, Broadway, NSW 2007, Australia; juri.hinzl@uts.edu.au

† Current address: School of Informatics and Data Science Hiroshima University 1-4-1 Kagamiyama, Higashi-Hiroshima City 739-8527, Japan.

Received: 16 September 2019; Accepted: 10 January 2020; Published: 17 January 2020

Abstract: Recently, diverse concepts originating from blockchain ideas have gained increasing popularity. One of the innovations in this technology is the use of the proof-of-work (PoW) concept for reaching a consensus within a distributed network of autonomous computer nodes. This goal has been achieved by design of PoW-based protocols with a built-in equilibrium property: If all participants operate honestly then the best strategy of any agent is also to follow the same protocol. However, there are concerns about the stability of such systems. In this context, the analysis of *attack vectors*, which represent potentially successful deviations from the honest behavior, turns out to be the most crucial question. Naturally, stability of a blockchain system can be assessed only by determining its most vulnerable components. For this reason, knowing the most successful attacks, regardless of their sophistication level, is inevitable for a reliable stability analysis. In this work, we focus entirely on blockchain systems which are based on the proof-of-work consensus protocols, referred to as PoW-based systems, and consider planning and launching an attack on such system as an optimal sequential decision-making problem under uncertainty. With our results, we suggest a quantitative approach to decide whether a given PoW-based system is vulnerable with respect to this type of attack, which can help assessing and improving its stability.

Keywords: blockchain; proof of work; distributed ledger; double-spending attack

1. Introduction

In recent years, concepts originating from the blockchain idea have gained popularity. Their software realizations are based on a mixture of traditional techniques (peer-to-peer networking, data encryption) and modern concepts (consensus protocols). Digital currencies represent assets of these systems, their transactions are written and kept in an electronic ledger as a part of operation of the blockchain system. Their main difference from a traditional financial system is that the assets (crypto-currencies) are not issued and supervised by a central authority, but by joint efforts of a network consisting of independent computers, all running the same/similar software. Such a network searches for consensus which yields a common version of the ledger shared by all participants. The consensus is reached in terms of a process, which is called *mining* and is usually backed by economic incentives. Proponent of blockchain systems argue that they can achieve the same level of certainty and security as those governed by a central authority at significantly lower costs. In fact, (the author thanks an anonymous referee), costs can be lower in some cases for the users, because of the lack of the service provider fees. However, in general, blockchain systems are more resource and energy consuming than centralized ones. Still, in return they provide decentralization that is the "splitting of trust" among a set of entities (possibly the entire network). Furthermore, due to the distributed, decentralized, and

homogeneous architecture of the network, a blockchain system can reach a high level of stability due to data redundancy and hard/software replicability.

Following a mining process, all network participants append, validate, and mutually agree on a common version of the data history, which is usually referred to as the *blockchain ledger*. Some authors consider the invention of mining as a real break-through which has solved a long-standing consensus problem in computer science, although this development must be considered in the context of notable research advances in consensus protocols like Byzantine fault tolerant protocols. There is also criticism of this approach. A critical point is that to reach a consensus, some real physical resources/efforts must be spent or at least allocated. For instance, the traditional Bitcoin protocol requires participants to solve cryptographic puzzles with real consumption of computing power and energy, in terms of the *proof of work* (PoW). Other blockchain systems avoid resource consumption and require a temporary allocation of diverse resources, for instance the ownership of the underlying digital assets (proof of stake), or their spending (proof of burn). Furthermore, commitment of storage capacity (proof of storage), or a diverse combination of resource allocation/consumption are also used.

Let us briefly elaborate on the proof of work whose details can be found in an excellent book by Andreas Antonopoulos: "Mastering Bitcoin" [1]. We focus on the Bitcoin protocol which was initiated by [2], with refinement on the double-spending problem by [3], and later in [4] with further considerations addressing propagation delay in [5]. In this framework, the ledger consists of a chain of blocks and each block contains valid transactions. The nodes compete to add a new block to the chain, and doing so, each node attempts to collect transactions and to solve a mathematical puzzle. Once this puzzle is solved, it is made public to other nodes. This protocol also prescribes that if a peer node reports a completed block, then it must be verified, and if this block is valid, it must be attached to the chain, all uncompleted blocks shall be abandoned and a new block continuing the chain must be started. However, even following these rules, the chain forks regularly, which results in different nodes working on different branches. To reach a consensus in such cases, the protocol prescribes that a branch with shorter length must be abandoned as soon as a longer branch becomes known.

Let us return to the stability of the PoW protocol in the sense of its resilience to attacks. Please note that within a blockchain system, the nodes are running a publicly available open-source software (for mining) which can easily be modified by any private user to control the computer nodes to undermine the system. In principle, there are many ways of doing this. One of the most obvious among malicious strategies would be an attempt to spend the electronic money more than once. The analysis of such a strategy is referred to as the double-spending problem.

In the classical [2,3] formulation of this problem it is suggested that a merchant waits for $n \in \mathbb{N}$ confirming blocks after a payment by the buyer, before a product/service is provided. While the network is mining these n blocks, the attacker tries to build his/her own secret branch containing a version of history in which this payment is not made. The idea is to not include the paying transaction into private secret branch. The attacker hopes that the private branch will overtake the official branch and will be incorporated into the long-term chain. If this strategy succeeds, then the private secret branch becomes official and the payment disappears in the ledger after the product/service is taken by the attacker. Nakomoto [2] provides and Rosenfeld [3] refines an estimate of the attacker's success probability depending on his/her computational power and the number n of confirming blocks.

Let us emphasize that [2,3] provide merely an idea why succeeding in the double-spending attack could be difficult since their analysis focuses on a simplified situation and lacks several important aspects. First, note that in the original work [3], the success estimation of the double spending is based on the assumption that the attacker can start the race having pre-mined one block. Still, it is not clear how to achieve an advantage of being able to start the race with one block ahead of the official chain. In fact, the present contribution is devoted to a systematic study of this interesting question.

Second, the work [2,3] merely calculates the probability of the secret chain getting ahead of the official one, ignoring the mining costs and revenues/losses from a successful/failed attack. Furthermore, the possibility of canceling secret mining (if the block difference becomes too high)

is not considered. Most important, however, is that it is assumed that the paying transaction must be placed right after the fork. Please note that this assumption is justifiable only if the merchant requires immediate payment after a purchase is agreed upon, otherwise canceling the deal. However, in reality, the attacker may be able to freely choose the time of payment, in particular when buying goods from web portals. That is, an attempt to overtake the official chain before launching an attack can give an advantage.

Remark 1. *In this work, we focus exclusively on PoW-based systems to analyze their vulnerability with respect to the double-spending threat, since other blockchain systems (all those based on different consensus algorithms, like permissioned networks) are immune to this type of attack. In this context, we examine the effect of pre-mining on the profitability of the double spending with two effects: Obviously, the success probability of the attack increases with the number of pre-mined blocks, while on the other hand, a longer mining race reduces rewards due to mining costs, i.e., whether the paying transaction must be placed immediately depends on the protocol's the block reward policy. Here, many technical details become crucial: For instance, ref. [6] investigates differences between Bitcoin and Ethereum with respect to rewards for stale and uncle blocks. However, such details are not covered by the present approach which elaborates on a general view and provides an algorithmic solution to the corresponding double-spending problem. Still, the code presented in this paper is flexible and can cover a wide range of situations, leaving enough space for specifications. For instance, there is an obvious linear relation between costs of secret mining and secret capacity fraction. In reality, this relation may be more complex, depending on mining hardware and its ownership. For this reason, we model mining costs and the capacity fraction with separate parameters leaving enough flexibility to tailor our implementation to a given problem, as illustrated in the Section 8 .*

Contribution of the paper: We discuss security assessment of double spending in terms of discrete-time finite-horizon stochastic control using an optimal stopping and a switching model. In contrast to an infinite-horizon discounted-reward Markov decision models suggested in the literature (see [6–8]), we obtain exact solutions and express our result via present-time monetary units which allows direct conclusions. In the optimal stopping formulation, we show how to choose the optimal payment moment depending on the length difference between the official and secret chains, mining capacity ratio, confirming block number, and on the revenue/loss from the success/failure of the attack. We upgrade this framework to a stochastic switching model and show how to decide whether it is worth attacking a given PoW-based system. This insight may allow important conclusions on its vulnerability. For all problems solved in this work, we provide a complete implementation and full source code listings which can be used for further adaptations.

Paper organization: Section 2 presents a literature review relevant to our paper, whereas Section 3 provides a motivation for our approach, which requires a finite-horizon framework introduced in Section 4. With methodological background from Section 5, we address the optimal stopping and switching models in Sections 6 and 7, whose implementation is illustrated by code listings and a number of numerical case studies in Section 8 with conclusions given in Section 9. The Appendix A is devoted to technical details.

2. Stochastic Models in the Analysis of Blockchains

Performance evaluation and improvement of blockchain systems relies on stochastic modeling, analysis, and optimization and is an active area with a substantial number of publications meanwhile. To analyze blockchain systems, diverse methods encompassing random walks, queuing models, Markov processes, stochastic control, and game theoretical models have been successfully applied. Let us mention some representative literature related to the *proof-of-work protocol*. Other consensus protocols (see [9]) are also discussed in terms of interesting models (for instance in [10]), but are outside of our scope. For a detailed overview of literature about applications of stochastic methods to blockchains, we refer the interested reader to the recent work [11].

Applications of queuing theory deal with modeling of transaction arrivals and block generation times and are discussed in [12,13]. An early and important work [14] on application of Markov processes to blockchains discusses vulnerability of the PoW protocol in the framework of the so-called selfish mining. More precisely, the work [14] suggests that a pool of miners may work secretly together to obtain higher payoffs than other miners violating the protocols by postponing block publishing. Such behavior is referred to as *selfish mining* whose investigation is extended, among others, by [5,15], the latter also considers propagation delay. The double-spending problem is discussed in [16] using renewal theory and in [17] using random walks.

The theory of Markov decision processes is applied to blockchain analysis in [6–8]. These publications are directly related to the present work. In [7], the idea of pre-mining on a secret branch invalidating a transaction for double spending is investigated. The authors recognize that if the payment moment can be chosen by the attacker, then the double spending succeeds with probability one. Using an infinite-horizon Markov decision formulation, the action to "adopt" (abandon secret mining) "override" (publish a longer secret chain), "match" (publish a secret chain of the same length), and "wait" (continue secret mining) are optimized in a specific framework. This study elaborates on the difference in communication to full nodes versus light nodes and its role for the success of the attack. A further refinement of this approach is suggested in [6]. This work optimizes a similar range of decisions for maximization of the proportion between secret and total mining rewards. Furthermore, the study [6] introduces an appropriate benchmark, defined as the minimal value of the double spending gain which makes the optimal selfish mining more profitable than the honest mining. Using this benchmark, diverse blockchain systems are compared and conclusions are derived. In Section 7, we discuss advantages and incremental contribution of our approach in relation to [6–8].

3. The Double-Spending Problem

Let us briefly discuss the classical results before we elaborate on our contribution. In the framework of the double-spending problem, it is assumed that a continuous-time Markov chain taking values in $\mathbb{Z}$ describes the difference in blocks between official and secret branches. As in [3], we consider this process at time points at which new block in one of the branches is completed, which yields a discrete-time Markov chain $(Z_t)_{t=0}^{\infty}$. Having started secret mining after the block including the attacker's payment (at block time $t = 0$, $Z_0 = 0$) the attacker considers the following situation: At each time $t = 1, 2, 3 \ldots$, a new block in one of the branches (official or secret) is found, the block difference changes

$$\mathbb{P}(Z_t = z + 1 \,|Z_{t-1} = z) = 1 - q,$$
$$\mathbb{P}(Z_t = z - 1 \,|Z_{t-1} = z) = q,$$

where $q \in]0, 1[$ is the ratio of the computational power controlled by the attacker to the total mining capacity. Let us agree on the generic case where the attacker controls a smaller part of the mining power $0 < q < 1/2$ than that controlled by honest miners. In this case, if at any block time $t = 0, 1, 2, \ldots$ the block difference is $z \in \mathbb{Z}$, then the probability $a_\infty(z, q)$ that the secret branch overtakes the official branch within an unlimited time after t is given by

$$a_\infty(z, q) = \mathbb{P}(\min_{u=0}^{\infty} Z_u < 0 \,|\, Z_0 = z) = \begin{cases} 1 & \text{if } z < 0, \\ (\frac{q}{1-q})^{z+1} & \text{otherwise.} \end{cases} \quad (1)$$

Furthermore, at a time when the n-th block in the official branch is mined, the probability that the attacker has mined $m = 0, 1, 2, \ldots$ blocks follows the *negative binomial distribution* whose distribution function is given by

$$F_{q,n}(m) = \sum_{j=0}^{m} \binom{n+j-1}{j} (1-q)^n q^j, \qquad m = 0, 1, 2, \ldots. \tag{2}$$

Both results (1) and (2) are combined in [3] to obtain the success probability of the double spending as follows: Consider the situation where the attack starts when the length difference between the official and the secret branches is $k \in \mathbb{Z}$ with $k \geq -n$. Consider first the situation that at the time the n-th block in the official chain is completed, the attacker has mined m blocks with $m - k > n$ in which case the secret branch can be published immediately. The probability of this event is given by

$$\sum_{m=n+k+1}^{\infty} \binom{n+m-1}{m} (1-q)^n q^m = 1 - F_{q,n}(n+k).$$

Next, consider the opposite event, assuming that when the n-th official block is completed, the attacker has not overtaken the official chain in which case $m - k \leq n$. In this case, the probability of winning the race is given by

$$\begin{aligned} \sum_{m=0}^{n+k} \binom{n+m-1}{m} (1-q)^n q^m a_\infty(n+k-m, q) = & \\ &= \left(\frac{q}{1-q}\right)^{1+k} \sum_{m=0}^{n+k} \binom{n+m-1}{m} (1-q)^m q^n \\ &= \left(\frac{q}{1-q}\right)^{1+k} F_{1-q,n}(n+k). \end{aligned}$$

Clearly, the total success probability of the double spending is given by the sum of its probabilities in both cases and equals to

$$\begin{aligned} &r_{q,n}(k) = 1 - F_{q,n}(n+k) + \left(\tfrac{q}{1-q}\right)^{1+k} F_{1-q,n}(n+k) \\ &\text{for } n \in \mathbb{N}, k \in \mathbb{Z}, k \geq -n, \text{ and } q \in]0, \tfrac{1}{2}[. \end{aligned} \tag{3}$$

For instance, consider the situation that the attacker starts the fork-off and launches the payment at the same time, then $k = 0$ and if the merchant waits for $n = 6$ confirming blocks, then the attacks succeeds with probabilities $r_{q,n=6}(k = 0)$ which are relatively small if the attacker controls a small part (six $q = 0.06$ or eight $q = 0.08$ percent) of the mining capacity:

$$r_{0.06,6}(0) = 0.00037, \qquad r_{0.08,6}(0) = 0.0025$$

As a result, waiting for six blocks after the payment has been considered to be secure in the sense that with realistic efforts, it is practically impossible to succeed with double spending.

Remark 2. *Please note that in the original work [3], the estimation of the double spending is based on the assumption that the attacker can start the race having pre-mined one block, i.e., $k = -1$. This leads to a different (see Figure 1) success probability*

$$r_{q,n}(-1) = 1 - F_{q,n}(n-1) + F_{1-q,n}(n-1). \tag{4}$$

Still, it is not clear how to achieve an advantage of being able to start the race with one block ahead of the official chain. In fact, the present contribution is devoted to a systematic study of this interesting question.

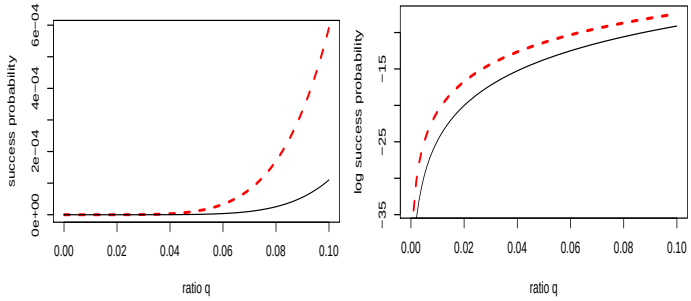

Figure 1. The success probability (and its logarithm) of the double-spending attack for $n = 6$ confirming blocks depending on the mining ratio $q \in [0, \frac{1}{10}]$ calculated by (3) with $k = 0$ (solid line) versus $k = -1$ as in (4) (dashed line).

The above analysis [3] calculates the probability of the alternative blockchain getting ahead of the official one. It does not consider revenues and losses from a successful/failed attack. Furthermore, the possibility of canceling the secret mining (if the block difference becomes too high) is not considered. Most important however, is the question why the paying transaction must be placed right after the fork. Please note that this assumption is justifiable only if the merchant requires immediate payment after the purchase is agreed upon, otherwise canceling the deal. However, in reality, the attacker may be able to freely choose the time of payment, in particular when buying goods from web portals. That is, an attempt to overtake the official chain before launching an attack can give an advantage in the spirit of the above remark.

4. Block Difference Dynamics

Consider a finite time horizon where $t \in \{0, \dots, T\}$ represents the number of blocks mined in the official chain since the branch has forked off, i.e., we suppose that our secret mining starts at the block time $t = 0$. We interpret $T \in \mathbb{N}$ as the maximal length of the official branch, which can be abandoned if a longer branch has been discovered. To the best of authors knowledge, the current Bitcoin protocol does not have such a restriction, meaning that the shorter branch must always be discarded, independently of its length. However, other systems discuss 'checkpoints' and 'gates' with similar functionality. A finite time horizon yields conceptual advantages (providing an exact solution) and presents a negligible deviation from reality since T is sufficiently large and can be changed in the calculation. Let us introduce all the ingredients required for a formal discussion of the decision problems formulated above. Introduce the block difference process $(Z_t)_{t=0}^T$,

$$\left.\begin{array}{c}\text{where } Z_t \text{ is the branch length difference between the official} \\ \text{and secret branch at times } t = 0, \dots, T \text{ when one new} \\ \text{block in the } \textit{official} \text{ branch is completed.}\end{array}\right\} \tag{5}$$

We show that the transition probabilities of $(Z_t)_{t=0}^T$ satisfy

$$\mathbb{P}(Z_{t+1} = x + 1 - j \,|\, Z_t = x) = \begin{cases} g(j) & j \in \mathbb{N} = \{0, 1, 2, 3, \dots\}, \\ 0 & j \in \mathbb{Z} \setminus \mathbb{N} = \{-1, -2, \dots\}, \end{cases} \tag{6}$$

for all $t = 0, \dots, T$ with a geometric distribution

$$g(j) = (1 - q)q^j \quad \text{for } j \in \mathbb{N} = \{0, 1, 2, 3, \dots\} \tag{7}$$

where $q \in [0,1]$ is the fraction of the capacity controlled by secret miners, the proof of the assertions (6) and (7) is found in Appendix A. Consider also $(\tilde{Z}_t = t - Z_t)_{t=0}^{T}$ with

$$\left.\begin{array}{c}\tilde{Z}_t \text{ is the length of the secret branch} \\ \text{at the times } t = 0, \ldots, T \text{ when one new} \\ \text{block in the } \textit{official} \text{ branch is completed.}\end{array}\right\} \tag{8}$$

This process possesses independent identically geometrically distributed increments

$$\mathbb{P}(\tilde{Z}_{t+1} = x + j \,|\, \tilde{Z}_t = x) = g(j) \qquad x, j \in \mathbb{N}, \;\; t = 0, \ldots, T. \tag{9}$$

In what follows, we show that determining the double-spending attack which has the highest expected total reward (from the viewpoint of the attacker) yields an optimal stochastic stopping/switching problem. To ease reader's understanding, we start with a simplified situation (neglecting secret mining costs, rewards for published blocks, and the possibility of abandoning the attack at any stage). Such a problem can be formulated as an optimal stopping problem. Thereafter, based on the setting of this stopping problem, we consider a more realistic approach and upgrade the optimal stopping to an optimal switching framework, which takes into account mining costs, rewards for published blocks, and the possibility to abandon secret mining. Before introducing all details in subsequent sections let us sketch the core ideas and explain how our results can be used to assess vulnerability of a given PoW-based system.

5. Decisions under Uncertainty: Optimal Switching and Stopping

Sequential decision-making arises in many applications and is usually addressed under the framework of discrete-time *Stochastic Control*. The theory of *Markov Decision Processes/Dynamic Programming* provides a variety of methods to deal with such questions. In generic situations, approaching solutions even for simplest decision processes may be a cumbersome process (ref. [18–20]). However, for the questions formulated in the present work, a specific truncation technique will be applied to state the problem on a finite space within a finite time horizon, which makes all results obtainable by finite number of algebraic operations at machine precision.

Let us introduce a particular Markov decision problem class: The *optimal stochastic switching* (see [21]). On a finite time horizon $\{0, 1, \ldots, T\}$ consider an agent concerned with the problem of sequential decision-making: At any time $t = 0, 1, \ldots, T-1$ an action $a \in A$ from a finite set A of all available actions must be chosen. This decision returns an immediate reward/costs but also influences the future state evolution, i.e., at any time, an action optimally balances between the current rewards/costs of control and all future situations. In the framework of optimal stochastic switching, the decision variable has two components $(p, z) \in E = P \times \mathbb{R}^d$ consisting of operation mode p and environment state z, thus the state space E is a Cartesian product of a finite set P all operation modes and the Euclidean space $\mathbb{R}^d$. Therefore, the evolution $(Z_t)_{t=0}^{T}$ of the second component is supposed to follow a Markov process with the interpretation that $Z_t = z$ is the situation in the global environment at time t which is relevant for making decisions but cannot be changed by an agent's actions. Contrary to this, the current operation mode $p \in P$ is under full deterministic control of the agent at any time. This aspect is modeled in terms of a function $\alpha : P \times A \to P$, $(p, a) \to \alpha(p, a)$, which describes a deterministic change of the operation mode by the agent's actions with the interpretation that $\alpha(p, a) \in P$ is the new mode if the action $a \in A$ was taken in the previous mode $p \in P$. Now, let us specify the control costs. Assume that taking an action $a \in A$ causes an immediate reward $r_t(p, z, a)$ which depends on the state $(p, z) \in E$ and on the action $a \in A$ through given reward functions $r_t : E \times A \to \mathbb{R}$ which may be time dependent. When the system arrives at the last time step $t = T$ in the state $(p, z) \in E$, the agent collects the *scrap value* $r_T(p, z)$, described by a pre-specified scrap function $r_T : E \to \mathbb{R}$. At each time $t = 0, \ldots, T-1$ the *decision rule* π_t is given by a mapping $\pi_t : E \to A$, prescribing at time t in the state $(p, z) \in E$ the action $\pi_t(p, z) \in A$. A sequence $\pi = (\pi_t)_{t=0}^{T-1}$ of decision

rules is called a *policy*. When controlling the system by the policy $\pi = (\pi_t)_{t=0}^{T-1}$, the positions $(p_t^\pi)_{t=0}^T$ and the actions $(a_t^\pi)_{t=0}^{T-1}$ evolve recursively

$$a_t^\pi = \pi_t(p_t^\pi, Z_t), \qquad p_{t+1}^\pi = \alpha(p_t^\pi, a_t^\pi), \qquad t = 0, \dots, T-1.$$

Having started at initial values $p_0^\pi = p_0$ and $Z_0 = z_0$, the goal of the controller is to maximize (over all possible policies) the expectation of the total reward

$$v_0^\pi(p_0, z_0) = \mathbb{E}\left(\sum_{t=0}^{T-1} r_t(p_t^\pi, Z_t, a_t^\pi) + r_T(p_T^\pi, Z_T)\right).$$

The function $(p, z) \mapsto v_0^\pi(p, z)$ is called the value of the policy π and represents the total reward accumulated within the entire time.

For technical details and solution algorithms to switching systems, we refer the interested reader to [22]. Furthermore, there are applications to pricing financial options [23], natural resource extraction [21], battery management [24] and optimal asset allocation under hidden state dynamics [25], many applications are illustrated using R in [26].

Let us now introduce the standard backward induction algorithm which is used to obtain a solution to an optimal switching problem. Given a switching problem as above, introduce the stochastic kernels for all $p \in P$, $a \in A$, $z \in \mathbb{R}^d$

$$\mathcal{K}_t^a v(p, z) = \mathbb{E}(v(\alpha(p, a), Z_{t+1}) \mid Z_t = z), \qquad t = 0, \dots, T-1, \tag{10}$$

which act on all functions v on $E = P \times \mathbb{R}^d$ where the above expectation exists. Using these kernels, the policy value is obtained recursively by the *policy valuation* algorithm

$$v_T^\pi = r_T, \quad v_t^\pi(p, z) = r_t(p, z, \pi_t(p, z)) + \mathcal{K}_t^{\pi_t(p,z)} v_{t+1}^\pi(p, z), \quad t =, T-1, \dots 0.$$

To obtain a policy $\pi^* = (\pi_t^*)_{t=0}^{T-1}$, which maximizes the total expected reward, one introduces for each $t = 0, \dots, T-1$ the so-called *Bellman operator*

$$\mathcal{T}_t v(p, z) = \max_{a \in A} \left(r_t(p, z, a) + \mathcal{K}_t^a v(p, z)\right), \quad (p, z) \in E \tag{11}$$

acting on each function $v : E \to \mathbb{R}$ where the above expectation exists. Now, consider the *Bellman recursion*, also referred to as the *backward induction*:

$$v_T = r_T, \quad v_t = \mathcal{T}_t v_{t+1}, \qquad \text{for } t = T-1, \dots, 0. \tag{12}$$

Under appropriate assumptions, there exists a recursive solution $(v_t^*)_{t=0}^T$ to the Bellman recursion

$$v_T^*(p, z) = r_T(p, z) \tag{13}$$

$$v_t^*(p, z) = \max_{a \in A} (r_t(p, z, a) + \mathbb{E}(v_{t+1}^*(\alpha(p, a), Z_{t+1}) \mid Z_t = z)) \tag{14}$$

for $t = T-1, \dots, 0$, $p \in P$, and $z \in \mathbb{R}^d$. The functions $(v_t^*)_{t=0}^T$ resulting from the backward induction are called *value functions*, they determine an optimal policy $\pi^* = (\pi_t^*)_{t=0}^{T-1}$ via

$$\pi_t^*(p, z) = \text{argmax}_{a \in A} \left(r_t(p, z, a) + \mathbb{E}(v_{t+1}^*(\alpha(p, a), Z_{t+1}) \mid Z_t = z)\right) \tag{15}$$

for $p \in P$, $z \in \mathbb{R}^d$, for all $t = 0, \dots, T-1$.

We shall emphasize that solutions to even simplest switching problems are sometimes surprising and non-intuitive. Frequently, observing an optimal solution helps to understand the original questions.

As an illustration, we consider two classical problems (borrowed from [18]) whose solutions are non-intuitive, at a first glance.

Game I: Consider a card desk face down with $b \in \mathbb{N}_+$ black and $r \in \mathbb{N}_+$ red cards. On each turn, the player chooses whether to draw a card from the desk or not. If the player decides to take a card, then he gains \$1 if a black card is taken, and loses \$1 if a red card is drawn. Once the card is taken, it is put aside and will not be returned to the desk. Is it possible that $b < r$ and it is still worth starting to draw?

Game II: An equal number of red and black cards $r = b \in \mathbb{N}_+$ are face down on a table. I am turning the cards over one by one, and at any time you can say "stop" and I turn over one next card. If the card is black, you win \$1, if it is red you lose \$1. If you do not stop before the last card, the last card's color is used to decide whether you win or lose. What is the optimal strategy for playing this game?

An analysis of the Game I shows that in some situations, it is indeed worth playing if there are more red than black cards. For instance, even for $b = 4$, $r = 6$ the value of the optimal policy is still positive, $2/300$. In a very similar Game II, it is surprisingly never worth playing since each policy returns the same value, which is zero.

The simplest and probably the most important special case of optimal stochastic switching is *optimal stopping*. Here, it is known that if the process $(Z_t)_{t=0}^T$ is stopped at $\tau = 0, 1, \ldots, T-1$, then the agent receives a value $R_\tau(Z_\tau)$ determined by a pre-specified *stopping reward* function $z \mapsto R_t(z)$, $t = 0, \ldots, T-1$, $z \in \mathbb{R}^d$. If the process is not stopped within $0, 1, \ldots, T-1$ then the agent receives $R_T(Z_T)$ determined by the given scrap function $(t, z) \mapsto R_T(z)$. The stopping problem is formulated as follows: Given $(Z_t)_{t=0}^T$ and $(R_t)_{t=0}^T$ as above, calculate the maximum and one of its maximizers to

$$\tau \mapsto \mathbb{E}(R_\tau(Z_\tau)) : \quad \text{where } \tau \text{ is a } \{0, 1, \ldots, T\}\text{-valued stopping time.}$$

Please note that the maximization is defined over stopping times, which comprise all random times not depending on future events. An optimal stopping problem can be equivalently formulated as an optimal stochastic switching problem. For this, define two positions and two actions $P = \{1, 2\}$, $A = \{1, 2\}$. Here, the positions "stopped" and "goes" are represented by $p = 1$, $p = 2$ respectively and the actions "stop" and "go" denoted by $a = 1$ and $a = 2$. With this interpretation, the position change is given by

$$(\alpha(p, a))_{p,a=1}^2 = \begin{bmatrix} \alpha(1,1) & \alpha(1,2) \\ \alpha(2,1) & \alpha(2,2) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}.$$

Please note that with this matrix the operation mode "goes" ($p = 2$, second row) remains valid only if the action "go" ($a = 2$, second column) is applied. If the systems is stopped ($p = 1$) or the action is to stop ($a = 1$) then the operation mode transitions to "stopped" ($p = 1$) and never leaves. The reward at time $t = 0, \ldots, T-1$ and the scrap value are defined by

$$r_t(p, z, a) = R_t(z)(p - \alpha(p, a)), \tag{16}$$

$$r_T(p, z) = R_T(z)(p - \alpha(p, 1)), \tag{17}$$

for all $p \in P$, $a \in A$, $z \in \mathbb{R}^d$.

For an optimal stopping problem, the backward induction can be written more compactly. Specifically, we introduce the value functions $(V_t)_{t=0}^T$ and the expected value functions $(V_t^E)_{t=0}^{T-1}$ recursively by

$$V_T^E = R_T, \quad V_t^E(z) = \mathbb{E}(V_{t+1}(Z_{t+1}) | Z_t = z), \quad V_t(z) = \max\{R_t(z), V_t^E(z)\}$$

for $t = T-1, \dots, 0$, $z \in \mathbb{R}^d$. The so-called continuation region is defined by

$$\mathcal{C} = \{(t,z) : t \in \{0, \dots, T-1\}, z \in \mathbb{R}^d : V_t^E(z) > R_t(z)\} \tag{18}$$

and the optimal stopping time τ^* is obtained as the first exit time of the process $(Z_t)_{t=0}^T$ from the region $\mathcal{C}$

$$\tau^* = \inf\{t = 0, \dots, T : (t, Z_t) \notin \mathcal{C}\}.$$

6. Attack Planning as an Optimal Stopping Problem

Assume that the attacker can freely choose the time of payment. Doing so, he/she can work on a private secret branch long before his/her payment is placed. For such situation, the analysis of the double spending is different from the approach explained in Section 3 and requires solving a stopping problem.

Consider the block difference dynamics $(Z_t)_{t=0}^T$ from (5). Launching the double-spending attack at the block time $\tau = 0, \dots, T$ simply means that the payment will be included into block $\tau + 1$ of the official branch. (Recall that the secret branch contains an invalidation of the paying transaction by a non-inclusion of the attacker's paying transaction into secret branch). That is, the crucial question is how to choose the block time $\tau = 0, \dots, T$ optimally at which the payment is made. Notice that although the state space $\mathbb{Z}$ of the Markov chain $(Z_t)_{t=0}^T$ is infinite, all relevant situations occur within a finite range. On this account, it is possible to formulate an equivalent optimal stopping/switching problem whose state process follows a finite-state and finite-horizon Markov chain. The idea is to appropriately adjust the original Markov dynamics to not leave a finite state range. For this, let us agree that

$$\left.\begin{array}{l} \{\tilde{Z}_t > t+n\} = \{Z_t < -n\}, \quad t = 0, 1, 2 \dots, \\ \text{represents a sure opportunity for a successful} \\ \text{double spending launched at time } \tau = t. \end{array}\right\} \tag{19}$$

Indeed, by attacking immediately with payment at $\tau = t$ if $\{Z_t < -n\}$ occurs, the last confirmation block is obtained at the time $t+n$, and the next official block is obtained at block time $t+n+1$ when the secret branch is at least of the same length $Z_{t+n+1} \leq Z_t + n + 1 \leq 0$ as the official chain, i.e., right before the block time $t+n+1$, the secret branch must have been longer.

Remark 3. *Please note that the insight (19) can be combined with the geometric distribution (9) to conclude that if the mining capacity ratio is positive, then the probability to succeed in the double-spending attack at least once in an infinite sequence of attempts equals to one. Indeed, suppose that $q > 0$. Having started secret mining, the probability of the event $\{\tilde{Z}_1 > 1+n\}$ that the attacker has more than $n+1$ blocks in the secret chain at the time when one official block has been completed is positive due to (9). However, if the attacker has not succeeded in overtaking $\{\tilde{Z}_1 \leq 1+n\}$, then the secret branch will be discarded, and a new chain bifurcation will be started, this time right after the current official block, with the attempt to overtake the official branch by more than $n+1$ at the time when the next official block is obtained. This second independent attempt yields a success with the same positive probability. Repeating this procedure, one obtains a sequence of Bernoulli experiments, each with the same positive success probability, which yields a success with probability one after a finite number of trials.*

Now, we clarify the relevant time horizon of the stopping problem. Because we have imposed a finite limit T on the length of the official branch, we agree that for $\tau > T - n$ a successful attack is not possible. Specifically, since the payment is placed into block $\tau + 1$ and n confirming blocks are expected, the last confirmation block $\tau + n > T$ would be beyond the maximal branch length which can be abandoned. That is, we can assume that the time τ must be chosen within the finite-horizon $\tau = 0, \dots, \tilde{T}$ with the last time point $\tilde{T} = T - n$. The decision whether to attack must be based on the current block time $t = 0, \dots, \tilde{T}$ and on the recent block difference Z_t.

The event that an attack launched at time $\tau = 0, \dots \tilde{T}$ is successful can be expressed in the form

$$S(\tau) = \{ \min_{i=\tau+n+1}^{T+1} Z_i \leq 0 \} \quad \tau = 0, \dots, \tilde{T}. \tag{20}$$

There is a "less than or equal to" in this expression since if at the block time $t = 1, \dots \tilde{T}$ the process has reached non-positive domain $Z_t \leq 0$, then immediately before the physical time corresponding to t, the block difference has been negative because at t one official block was completed (block difference increased at t to $Z_t \leq 0$).

In the second step, we define the stopping reward function as

$$\begin{aligned} R_\tau(x) &= \mathbb{E}(C1_{S(\tau)} - c1_{S(\tau)^c} \,|\, Z_\tau = x) \\ &= (C + c)\mathbb{P}(S(\tau)|\, Z_\tau = x) - c, \quad \tau = 0, \dots, \tilde{T}, \;\; x \in \mathbb{Z}, \end{aligned} \tag{21}$$

where the numbers $C > 0$ and $c > 0$ represent the gain and the loss resulting from the success or failure of the attack, respectively. Finally, let us agree that $\tau = \tilde{T} + 1$ stands for the attacker's option to not launch any attack, which can be optimal if the chance of overtaking the official branch is too low, in view of a potential loss from an unsuccessful attack. To model such an opportunity, we define the scrap function for the time argument $t = \tilde{T} + 1$ as

$$R_{\tilde{T}+1}(x) = 0, \quad x \in \mathbb{Z}. \tag{22}$$

Having introduced all ingredients, the choice of the attack time τ^* yields the double-spending problem in the optimal stopping formulation:

$$\left.\begin{array}{c} \text{determine the maximum and a maximizer to } \mathcal{T} \to \mathbb{R}, \;\; \tau \mapsto \mathbb{E}(R_\tau(Z_\tau)) \\ \text{where } \mathcal{T} \text{ denotes all } \{0, \dots, \tilde{T} + 1\} \text{ valued stopping times.} \end{array}\right\} \tag{23}$$

The next section deals with a solution to this problem. Like almost all stopping questions, our double-spending problem (23) is solved in terms of a recursive algorithm rather by an explicit formula. That is, investigating its solution structure requires numerical experimentation, thus parameter dependence of the optimal strategy is not obvious. Hence, we include a solution code, implemented in R.

From the numerical experiments conducted so far, the authors observed that the solution is natural, intuitive, and not surprising. Specifically, in all calculations we determine the same behavior: The only optimal strategy is to follow secret mining without launching an attack until the block difference reaches or exceeds a critical value which depends on model parameters. For instance, the optimal attack is triggered when/if the secret branch overtakes the official branch by two or more blocks (we illustrate this by an example). In all experiments, we also observed that the optimal strategy is time-homogeneous (the block difference, triggering the attack does not depend on the length of the official branch).

Remark 4. *Let us explain how to interpret these outcomes, derive conclusions and elaborate on what has been gained compared to existing results.*

- *First, our analysis shows that under a (realistic) assumption that the payment moment can be chosen by the attacker, estimating success probability of a double-spending attack an ill-posed question. Specifically, since the increments of the branch length difference follow a geometric distribution (6), the attacker will succeed with probability one, by simply repeating over and over again the chain bifurcation, any time after having been overtaken by the official branch. Please note that this argument applies to a repeated sequence of attempts of arbitrary length rather to a single attempt, as pointed out in the remark after (19).*

- *Second, the only reason such a strategy is not profitable are the costs of private mining related to the potential gain/loss from a successful/unsuccessful attack and their probabilities. These economic aspects are crucial and, in difference to the previous work, are reflected in our approach by the gain/loss parameters C and c along with quantified success/failure probability of the double-spending.*
- *Third, our results can be used for vulnerability assessment (the author is extremely grateful to an anonymous referee for suggestions, which helped addressing PoW stability in terms of optimal switching techniques) of a given PoW-based system. However, for this we must include beyond gain/loss parameters and mining capacity relation also further details: Costs of mining, rewards for published blocks and the option to abandon secret mining at any time, resulting in three operation modes, rather than two in the optimal stopping case. This yields a more complex model. We shall sketch such an approach in the following section.*

6.1. Attack Optimization in the Optimal Stopping Formulation

In our numerical approach we use the length of the secret chain $(\tilde{Z}_t)_{t=0}^{\tilde{T}+1}$ from (8) as our state process. According to the observation (19), the evolution of the underlying state process needs to be examined merely on a finite range

$$\{(t,x) : t = 0,1,\ldots,\tilde{T}+1, x = 0,\ldots,t+n\} \subset \mathbb{N}. \tag{24}$$

Having re-defined the reward (21) in accordance to (19) and (22) as

$$\tilde{R}_t(x) = \begin{cases} R_t(t-x) & \text{for } t = 0,\ldots,\tilde{T}, \quad x = 0,\ldots,t+n, \\ C & \text{for } t = 0,\ldots,\tilde{T}, \quad x = t+n+1,\ldots, \\ 0 & \text{for } t = \tilde{T}+1, \quad x \in \mathbb{N}, \end{cases} \tag{25}$$

we equivalently re-formulate the problem (23) as

$$\begin{gathered} \text{determine a maximizer } \tau^* \text{ to } \mathcal{T} \to \mathbb{R}, \quad \tau \mapsto \mathbb{E}(\tilde{R}_\tau(\tilde{Z}_\tau)) \text{ where} \\ \mathcal{T} \text{ denotes all } \{0,\ldots,\tilde{T}+1\}\text{-valued stopping times.} \end{gathered} \tag{26}$$

To solve the above optimal stopping problem, we introduce the value functions $(\tilde{V}_t)_{t=0}^{\tilde{T}}$ to (26) in terms of the standard backward induction, which is initialized by the expected value function

$$\tilde{V}_{\tilde{T}}^E(x) = 0, \qquad x \in \mathbb{N}, \tag{27}$$

and is followed recursively for $t = \tilde{T}, \tilde{T}-1,\ldots,$ by

$$\tilde{V}_t(x) = \max\{\tilde{R}_t(x), \tilde{V}_t^E(x)\} \quad x \in \mathbb{N}, \tag{28}$$

$$\tilde{V}_{t-1}^E(x) = \mathbb{E}(\tilde{V}_t(\tilde{Z}_t)|\tilde{Z}_{t-1} = x), \quad x \in \mathbb{N}. \tag{29}$$

Since $\tilde{V}_t(x) = C$ for all $x > t+n$, each value function $\tilde{V}_t(x)$ needs to be calculated only for states $x = 0,1,\ldots,t+n$. We thus obtain instead of (27)–(29)

$$\begin{aligned} \tilde{V}_{\tilde{T}}^E(x) &= 0, \qquad x = 0,\ldots,\tilde{T}+n, \\ \tilde{V}_t(x) &= \max\{\tilde{R}_t(x), \tilde{V}_t^E(x)\} \quad x = 0,\ldots,t+n, \\ \tilde{V}_{t-1}^E(x) &= \mathbb{E}(\tilde{V}_t(\tilde{Z}_t)|\tilde{Z}_{t-1} = x) \quad x = 0,\ldots,t+n-1. \end{aligned}$$

Please note that in the last equality, the conditional expectation can be calculated as

$$\begin{aligned}\tilde{V}^E_{t-1}(x) &= \sum_{j=0}^{t+n-x} \tilde{V}_t(x+j)g(j) + C\sum_{j=t+n+1-x}^{\infty} g(j)\\ &= (1-q)\sum_{j=0}^{t+n-x} \tilde{V}_t(x+j)q^j + Cq^{t+n+1-x}, \quad x = 0,\dots,t+n-1.\end{aligned}$$

Having determined the value functions $\tilde{V}_t(x)$ for $t = 0,\dots,\tilde{T}$ and $x = 0,\dots,t+n$, continuation region is obtained by

$$\mathcal{C} = \{(t,x) : t \in \{0,\dots,\tilde{T}\}, x \in \{0,1,\dots,t+n\} : \tilde{V}^E_t(x) > \tilde{R}_t(x)\}$$

and the optimal attack time τ^* is obtained as the first exit time of the process $(\tilde{Z}_t)_{t=0}^{\tilde{T}}$ from the region $\mathcal{C}$

$$\tau^* = \inf\{t = 0,\dots,\tilde{T}+1 : (t,\tilde{Z}_t) \notin \mathcal{C}\}.$$

6.2. Algorithmic Solution

Before we present an algorithmic solution, let us show how to calculate the rewards (21). In order to determine the probability $\mathbb{P}(S(t)|\,Z_t = x)$ in the expression (21), we use the time and space homogeneity of the transition kernel to obtain

$$\begin{aligned}&\mathbb{P}(S(t)|\,Z_t = x) = \mathbb{P}(\min_{i=n+1}^{T+1-t} Z_i \le 0\,|\,Z_0 = x) \qquad (30)\\ &= \sum_{x'=1}^{x+n+1} \mathbb{P}(\min_{i=n+1}^{T+1-t} Z_i \le 0\,|\,Z_{n+1} = x')\mathbb{P}(Z_{n+1} = x'\,|\,Z_0 = x) + \mathbb{P}(Z_{n+1} \le 0\,|\,Z_0 = x)\\ &= \sum_{x'=1}^{x+n+1} \mathbb{P}(\min_{i=0}^{T-t-n} Z_i \le 0\,|\,Z_0 = x')\mathbb{P}(Z_{n+1} = x'\,|\,Z_0 = x) + \mathbb{P}(Z_{n+1} \le 0\,|\,Z_0 = x).\end{aligned}$$

To calculate the probabilities in this expression, let us consider a truncation of the dynamics $(Z_t)_{t=0}^T$ by making upper and lower ranges of the state space absorbing. Specifically, given the lower and upper boundaries $l, u \in \mathbb{Z}$ in the state space, consider an alternative Markovian dynamic $(Z_t^{(l,u)})_{t=0}^T$ on the truncated state space $\{l-1,\dots u+1\} \subset \mathbb{Z}$ whose transition matrix

$$p^{(l,u)} = (p^{(l,u)}_{x,x'} = \mathbb{P}(Z^{(l,u)}_{t+1} = x'|\,Z_t = x))_{x,x'=l-1}^{u+1}, \qquad t = 0,\dots,T$$

is obtained from the transition matrix

$$p = (p_{x,x'} = \mathbb{P}(Z_{t+1} = x'|\,Z_t = x))_{x,x'\in\mathbb{Z}} \qquad t = 0,\dots,T$$

by the truncation procedure:

$$\begin{aligned}&p^{(l,u)}_{i,j} = p_{i,j} \text{ for } l \le i,j \le u, \quad p^{(l,u)}_{u-1,u-1} = p^{(l,u)}_{l+1,l+1} = 1,\\ &p^{(l,u)}_{i,u+1} = \textstyle\sum_{j>u} p_{i,j}, \quad p^{(l,u)}_{i,l-1} = \sum_{j<l} p_{i,j} \text{ for } l \le i \le u.\end{aligned}$$

Please note that the evolution of $(Z_t^{(l,u)})_{t=0}^T$ coincides with that of $(Z_t)_{t=0}^T$ on all states x with $l \le x \le u$ but as soon as $(Z_t)_{t=0}^T$ leaves this area, the dynamics becomes trapped in the lower $l-1 \in \mathbb{Z}$ or in the upper $u+1 \in \mathbb{Z}$ state depending on which boundary l or u has been crossed. Using this truncation technique, we obtain the required probabilities explicitly.

Lemma 1.

(a) Suppose that $x \in \mathbb{Z}, n \in \mathbb{N}$ with $x \geq -n$, then for $l, u \in \mathbb{Z}$ with $l \leq -n$ and $x + n + 1 \leq u$

$$\mathbb{P}(Z_{n+1} \leq 0 \,|\, Z_0 = x) = \sum_{y=l-1}^{0} (p^{(l,u)})^{n+1}_{x,y}, \tag{31}$$

$$\mathbb{P}(Z_{n+1} = x' \,|\, Z_0 = x) = (p^{(l,u)})^{n+1}_{x,x'} \quad \text{for } x' = 1, \ldots, x + n + 1. \tag{32}$$

(b) If $x', k \in \mathbb{N}_+$ then with $l = 0$ and $k + x' \leq u$

$$\mathbb{P}(\min_{i=0}^{k} Z_i \leq 0 \,|\, Z_0 = x') = (p^{(l,u)})^{k}_{x',0}.$$

The proof of this lemma is found in Appendix A.

Let us outline the use of the above lemma for determining the conditional probabilities (30) on a range of relevant states $x \in \{-n, -n+1, \ldots, x_{\max}\}$ for $T - t \geq n$. First, consider the $n+1$-step transition probabilities. Using (32), we conclude that with $l = -n$ and $u = x_{\max} + n + 1$ we obtain

$$\mathbb{P}(Z_{n+1} = x' \,|\, Z_0 = x) = (p^{(l,u)})^{n+1}_{x,x'} \quad \text{for } x \in \{-n, \ldots, x_{max}\}, x' \in \{1, \ldots, x + n + 1\}.$$

Using the space homogeneity of the transition kernel (6), we shift all states and boundaries by $n + 2$ to ensure that with $l = 2$ and $u = 2n + x_{\max} + 3$

$$\mathbb{P}(Z_{n+1} = x' \,|\, Z_0 = x) = (p^{(l,u)})^{n+1}_{x+n+2,x'+n+2} \quad x \in \{-n, \ldots, x_{max}\}, x' \in \{1, \ldots, x + n + 1\}.$$

Similarly, with the same boundaries $l = 2$ and $u = 2n + x_{\max} + 3$ we obtain for all $x \geq -n$

$$\mathbb{P}(Z_{n+1} \leq 0 \,|\, Z_0 = x) = \sum_{y=1}^{n+2} (p^{(l,u)})^{n+1}_{x+n+2,y}.$$

Moreover, given $k \in \mathbb{N}$, for the boundaries with $l = 2$ and $u = k + x'_{\max} + 1$, we obtain for all $x' \in \{1, \ldots, x'_{\max} = x_{max} + n + 1\}$

$$\mathbb{P}(\min_{i=0}^{k} Z_i \leq 0 \,|\, Z_0 = x') = (p^{(l,u)})^{k}_{x'+1,1}.$$

In order to calculate the conditioned probability (30) for $t \in \{0, \ldots, \tilde{T}\}$ and $x = -n, \ldots, t$, the above truncation technique yields

$$w^{\tilde{T},t}_{q,n}(x) = \sum_{x'=1}^{x+n+1} Q_{x'+1,1} P_{x+n+2,x'+n+2} + \sum_{y=1}^{n+2} P_{x+n+2,y}, \tag{33}$$

where the matrices P and Q are obtained by setting $x_{max} = t$ and $k = \tilde{T} - t$ in

$$P = (p^{(2,2n+x_{max}+3)})^{n+1} = (p^{(2,2n+t+3)})^{n+1}, \tag{34}$$

$$Q = (p^{(2,k+x_{max}+n+1)})^{k} = (p^{(2,k+t+n+1)})^{\tilde{T}-t}. \tag{35}$$

Please note that with the function (33)

$$\mathbb{P}(S(t) |\, Z_t = x) = w^{\tilde{T},t}_{q,n},(x) \quad \text{for } t = 0, \ldots, \tilde{T} \text{ and } x = -n, \ldots, t$$

which shows that for large $\tilde{T} = T - n$

$$w_{q,n}^{\tilde{T},t}(x) \approx r_{q,n}(x) \quad \text{for } t = 0, \dots, \tilde{T} \text{ and } x = -n, \dots, t. \tag{36}$$

Having calculated (30) in this way, the reward (25) is obtained for $t = 0, \dots, \tilde{T}$ by

$$\tilde{R}_t(x) = (C + c) w_{q,n}^{\tilde{T},t}(t - x) - c \quad \text{for } x = 0, \dots, t + n. \tag{37}$$

These results can be combined to formulate an algorithm that calculates the optimal stopping time and the optimal value function:

- **Step 1:** Initialize the backward induction by

$$\tilde{V}_{\tilde{T}}^{E}(x) = 0 \quad \text{for } x = 0, \dots, \tilde{T} + n, \text{ set } t := \tilde{T}.$$

- **Step 2:** Given $t \in \{0, \dots, \tilde{T}\}$

  (a) For $x = 0, \dots, t + n$ define

$$\tilde{R}_t(x) = (C + c) w_{q,n}^{\tilde{T},t}(t - x) - c.$$

  (b) Define $\tilde{V}_t(x) = \max(\tilde{R}_t(x), \tilde{V}_t^E(x))$ for $x = 0, \dots, t + n$.

  (c) Determine the conditional expectation V_{t-1}^E of the value function $\tilde{V}_t$ by

$$\begin{gathered} \tilde{V}_{t-1}^{E}(x) = (1 - q) \textstyle\sum_{j=0}^{t+n-x} \tilde{V}_t(x + j) q^j + C q^{n+t-x+1}, \\ \text{for all } x = 0, \dots, t + n - 1. \end{gathered} \tag{38}$$

  If $t > 0$, then repeat the Step 2 with $t := t - 1$. Otherwise, if $t = 0$, finish.

Section 8.1 illustrates this algorithm.

7. Attack Planning as an Optimal Switching Problem

Given the number $n = 1, 2, \dots$ of required confirmations, we consider $n + 2$ operation modes $p \in P = \{1, 2, \dots, n + 2\}$ which are interpreted as $p = 1$ "mining abandoned", $p = 2$ "mining continues", $p = k + 2$ "attack launched and confirmation block $k = 1, \dots, n$ completed". Introduce three actions $a \in A = \{1, 2, 3\}$ with switching matrix

$$(\alpha(p,a))_{p,a=1}^{n+2,3} = \begin{bmatrix} \alpha(1,1) & \alpha(1,2) & \alpha(1,3) \\ \alpha(2,1) & \alpha(2,2) & \alpha(2,3) \\ \alpha(3,1) & \alpha(3,2) & \alpha(3,3) \\ \vdots & \vdots & \vdots \\ \alpha(n+2,1) & \alpha(n+2,2) & \alpha(n+2,3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 4 & 4 \\ \vdots & \vdots & \vdots \\ 1 & n+2 & n+2 \end{bmatrix} \tag{39}$$

whose interpretation is context-dependent and is given as follows: If secret mining is abandoned (mode $p = 1$), then there is no return to any other mode. If the attacker mines on a secret chain without having launched an attack yet (mode $p = 2$, second row), then the mining can be abandoned by $a = 1$ as $\alpha(2,1) = 1$, continued by $a = 2$ as $\alpha(2,2) = 2$, or the attack can be launched by $a = 3$ as $\alpha(2,3) = 3$. If the attack is already launched and $k = 1, \dots, n$ confirmation blocks are received (mode $p = k + 2$) then there are two possibilities: either to continue secret mining by $a = 2, 3$ as $\alpha(p,2) = \alpha(p,3) = (p + 1) \wedge (n + 2)$ (giving next confirmation block) or to abandon by $a = 1$ as $\alpha(k,1) = 1$.

Next, let us use the secret branch length $(\tilde{Z}_t)_{t=0}^{T+1}$ from (8) as a state process and introduce control costs as function of this state. If mining is abandoned ($p = 1$), then there are no costs

$$r_t(1, z, a) = 0, \quad \text{for all } z \in \mathbb{N}, a \in A, t = 0, \ldots, T.$$

If mining continues ($p = 2$), then the mining costs $m \geq 0$ must be paid, thus

$$r_t(2, z, 2) = r_t(2, z, 3) = -m, \qquad \text{for all } z \in \mathbb{N}, t = 0, \ldots, T.$$

In this mode $p = 2$, abandoning secret mining by $a = 1$ has two interpretations. If the attacker is ahead of the official chain ($\tilde{Z}_t = z > t$), then the secret chain will be published and the attacker receives a reward $\rho \geq 0$ for all blocks mined so far

$$r_t(2, z, 1) \quad = \quad \rho \cdot z 1_{\{z>t\}}, \quad \text{for all } z \in \mathbb{N}, t = 0, \ldots, T.$$

Similarly, if the attack is launched ($p > 2$) then again, mining costs must be paid:

$$r_t(p, z, 2) = r_t(p, z, 3) = -m, \qquad z \in \mathbb{N}, \quad t = 0, \ldots, T.$$

In this mode ($p > 2$) abandoning secret mining has again two interpretations. If the attacker is ahead of the official chain ($\tilde{Z}_t = z > t$), then the secret chain will be published and the attacker receives a reward for each block mined so far. Furthermore, if the official chain was overtaken and at least n confirmation blocks are received (which corresponds to $\tilde{Z}_t = z > 0$ and $p = 2 + n$), then also a revenue $C > 0$ from a successful double spending is collected. However, if there are not enough confirmation blocks ($2 < p < 2 + n$) then the attack was unsuccessful which causes a loss $c > 0$:

$$r_t(p, z, 1) \quad = \quad \rho \cdot z 1_{\{z>t\}} + C 1_{\{z>t\}} 1_{\{p=2+n\}} - c 1_{\{2<p<2+n\}}, \quad z \in \mathbb{N}, t = 0, \ldots, T.$$

At the end $t = T + 1$ of the time horizon, if an attack has been launched but the secret chain was not published ($2 < p < 2 + n$), then the attack was unsuccessful which yields a loss $c > 0$:

$$r_{T+1}(p, z) = -c 1_{\{2<p\leq 2+n\}}, \quad \text{for all } z \in \mathbb{N}, p \in P.$$

With the above specifications, we introduce the double-spending problem in the optimal switching formulation as follows:

$$\left.\begin{array}{c} \text{given } (p_0, z_0) \in E \text{ determine the maximum and a maximizer to} \\ \pi \mapsto v_0^\pi(p_0, z_0) = \mathbb{E}\left(\sum_{t=0}^{T} r_t(p_t^\pi, \tilde{Z}_t, a_t^\pi) + r_{T+1}(p_{T+1}^\pi, \tilde{Z}_{T+1})\right) \\ \text{over all control policies } \pi. \end{array}\right\} \tag{40}$$

Recall that the system starts with chain bifurcation ($z_0 = 0$) by secret mining (in the mode $p_0 = 2$). Once the optimal strategy π^* from (40) is determined, the PoW vulnerability can be assessed in terms of the optimal policy value at this point. Specifically,

$$\left.\begin{array}{c} \text{if } v_0^{\pi^*}(p_0, z_0) = v_0^{\pi^*}(2, 0) = 0, \text{ then there} \\ \text{is no profitable double-spending attack} \end{array}\right\} \tag{41}$$

since π^* yields the same gain/loss as abandoning secret mining immediately.

Remark 5. *In practice, assessing PoW stability may require more complex considerations than solving (40). As mentioned earlier, secret miners can slow down honest mining: The idea (see [14] (the authors thank to an anonymous referee), refs. [6–8]) that once ahead of the official chain, secret miners can reveal blocks from their private branch to the public such that the honest miners switch to the recently revealed blocks, abandoning their*

shorter public branch. This strategy leads honest miners to waste resources working on blocks that are already mined, in the sense that they are working on the secret chain and are behind of secret miners.

Remark 6. *Selfish mining can be considered to be an attack with the purpose of obtaining larger rewards for mining than that of the honest pool, or to dominate the mining capacity for government of the network. In some sense, selfish mining can be considered to be a part of our optimal double-spending problem due to revenue from the secretly mined blocks. However, we do not model a strategy for secret block publications, thus the core mechanism of selfish mining is not included in the preset approach. These aspects should be considered to further refine the double-spending analysis.*

Attack Optimization in the Optimal Switching Formulation

Given the state process $(\tilde{Z}_t)_{t=0}^{T+1}$ from (8) and switching matrix (39) the optimal control problem (40) is solved via backward induction (13), (14). However, (14) requires determining an expectation with respect to a geometric distribution, which involves an infinite number of summations. Still, this calculation can be reduced to a finite number of operations since our value functions are constant, starting from a sufficiently large state variable. More precisely, we verify below that our assumption that there is a maximal chain length T which can be abandoned implies that $\nu_t^*(p,z) = \nu_t^*(p,T+1)$ for all $z \in \mathbb{N}$ with $z > T$. Specifically, the value functions $\nu_t^*(p,z)$ of the optimal policy can be explicitly calculated for large values $z > T$ as

$$\begin{aligned} \nu_t^*(p,z) &= 1_{\{1<p\}}\rho(T+1) + 1_{\{2<p\}}(C1_{\{t\le T-n+p-2\}} - c1_{\{t>T-n+p-2\}}) \\ &\quad +1_{\{p=2\}}C1_{\{t\le T-n\}} \quad \text{for all } z \in \mathbb{N}, z > T, p \in P, t = 0,\dots,T. \end{aligned} \tag{42}$$

Indeed, if the secret chain exceeds $z > T$ the maximal branch length which can be abandoned then further mining is not profitable and the gain from publishing a longer branch is $1_{\{1<p\}}\rho(T+1)$. Furthermore, having a secret branch of this length $z > T$, there is a good chance of the double-spending attack succeeding. For this, the publication of the chain should be postponed until the last confirmation block is received. Please note that the number of confirmation blocks required is $n-(p-2)$ whereas $T-t$ is the number of official blocks to be received until the time horizon ends. Suppose that the attack is already launched $p > 2$ then it succeeds if $n-(p-2) \le T-t$ with the publication of the secret branch (until block $T+1$) after the official block $t+n-(p-2) \le T$, immediately after the last confirmation block is received. This explains the term $C1_{\{t\le T-n+p-2\}}$ in the expression (42). Otherwise, if $n-(p-2) > T-t$ then the last confirmation block arrives after the official block T, thus the attack does not succeed which gives a loss term $-c1_{\{t>T-n+p-2\}}$ in the formula (42). If the network is not attacked yet $p = 2$, then the attack shall be launched immediately if $n \le T-t$, otherwise there will be no attack. This ensures that the last confirmation block is received at $t+n \le T$ before the end of the time horizon, giving the gain term $1_{\{p=2\}}C1_{\{t\le T-n\}}$ in (42).

Let us summarize the control costs formulated in Section 7 as

$$r_t(p,z,1) = \rho \cdot z1_{\{z>t\}}1_{\{1<p\}} + C1_{\{z>t\}}1_{\{p=2+n\}} - c1_{\{2<p<2+n\}}, \tag{43}$$

$$r_t(p,z,2) = r_t(p,z,3) = -m1_{\{1<p\}}, \tag{44}$$

$$r_{T+1}(p,z) = -c1_{\{2<p\le 2+n\}}, \tag{45}$$

for $t = 0,\dots,T$, $p \in P$ and $z \in \mathbb{N}$ and provide a solution to the stochastic switching problem (40) in terms of the following algorithm:

- **Step 1:** Calculate the expected value function using scrap values from (45):

$$\nu_{T+1}^E(p,z) = r_{T+1}(p,z) \quad \text{for } p \in P, z = 0,\dots,T, \text{ set } t := T.$$

- **Step 2:** Given $t \in \{0,\dots,T\}$

(a) Use (43) and (44) to calculate $r_t(p,z,a)$ for $p \in P$, $a \in A$ and $z = 0,\ldots,T$.

(b) Use switching matrix α from (39) to determine

$$\begin{aligned} v_t(p,z) &= \max_{a\in A}(r_t(p,z,a) + v_t^E(\alpha(p,a),z)) \\ &\quad \text{for } z = 0,\ldots,T,\ p \in P,\ a \in A. \end{aligned} \tag{46}$$

(c) Calculate the expected value functions $v_{t-1}^E(p,z)$ for $p \in P$ and $z = 0,\ldots,T$

$$v_{t-1}^E(p,z) = (1-q)\sum_{j=0}^{T-z} v_t(p,z+j)q^j + v_t^*(p,T+1)q^{T-z+1} \tag{47}$$

where $v_t^*(p,T+1)$ results from (42). If $t = 0$ then finish, otherwise repeat Step 2 with $t := t-1$.

We provide a numerical illustration of this algorithm in Section 8.2.

Remark 7. *Our approach provides several advantages compared to the framework of infinite-horizon Markov decision approach applied in [6–8] due to following aspects: Using a finite time horizon, we consider a wider policy class than in the infinite-horizon discounted-reward approach (all policies instead of those which are stationary). Furthermore, we obtain exact solutions by a finite number of algebraic operations (rather than relying on convergence). As a result, all our policy values are expressed exactly in present-time monetary units since there is no artificial discounting. Please note that unlike in the infinite-horizon discounted-reward setting, monetary policy values allow direct conclusions since there is no need for comparison and benchmarking. Finally, using a finite number of operational switching modes yields a compact and natural problem description with few actions and a relatively small state space. Nevertheless, Markov decision models (particularly those from [6]) address and manage many technical details using existing Markov decision solvers.*

8. Experimental Results

8.1. Numerical Illustration of Optimal Stopping

Let us illustrate the above algorithm by an implementation in the scientific computing language R. We define all auxiliary functions required by (33) (matrices P and Q in (34) and (35))

```
rm(list = ls()) # remove all objects
library(expm) #install.packages("expm")
make_matrix<-function(q, d) # routine  to generate matrices
{                           # required by the function w()
stopifnot(d>=2)
mat<-matrix(data=0, nrow=d, ncol=d)
mat[d,d]<-mat[1,1]<-1
for (i in 2:(d-1)) {
for (j in 2:(i+1)) mat[i,j]<-(1-q)*q^(i+1-j)
mat[i, 1]<-q^(i)
}
return(mat) }
```

and the transition matrix of $(\tilde{Z}_t)_{t=0}^{\tilde{T}}$ required by (38).

```
make_tr_matrix<-function(q, d) # function to generate matrix
{                              # required by conditional expectation
stopifnot(d>=2)
mat<-matrix(data=0, nrow=d-1, ncol=d)
for (i in 1:(d-1)) {
for (j in i:(d-1)) mat[i,j]<-(1-q)*q^(j-i)
mat[i, d]<-q^(d-i)
}
return(mat) }
```

Next, implement (33) and its approximation (36) based on (3) to define the reward function:

```
win_race_1<-function(q, n, t, x, tildeT)  # function w()
{stopifnot((0<=t)&(t<=tildeT)&(-n <=x)&(x<=t))
P<-make_matrix(q=q, d=2*n +t +4)%^%(n+1)
Q<-make_matrix(q=q, d=tildeT+n+2)%^%(tildeT-t)
index1<-1:(x+1+n)
index2<-1:(n+2)
prob1<-sum(P[x+n+2, index1 +n+2]); prob2<-sum(P[x+n+2, index2])
stopifnot(abs(prob1+prob2-1)<0.000000001)
result<-sum(Q[index1+1, 1]*P[x+n+2, index1 +n+2] )+sum(P[x+n+2, index2])
return(result) }
############################
win_race_2<-function(q, n, t, x, tildeT) # fast approximation to  w()
{stopifnot((0<=t)&(t<=tildeT)&(-n <=x)&(x<=t))
1-pnbinom(q=n+x, size=n+1, prob=1-q)+
((q/(1-q))^(1+x))*pnbinom(q=n+x, size=n+1, prob=q) }
############################
make_reward<-function(t, tildeT, n, q, C, c)
{ stopifnot((0<=t)&(t<=tildeT)&(0<=C)&(0<=c)&(0<=n)&(0<q)&(q<=0.5))
result<-rep(0, t+n+1)
if (app==TRUE)
for (x in 0:(t+n))  result[x+1]<-
win_race_2(q=q, n=n,t=t, x=t-x, tildeT=tildeT) else
for (x in 0:(t+n)) result[x+1]<-
win_race_1(q=q, n=n,t=t, x=t-x, tildeT=tildeT)
result<-(C+c)*result-c
return(result) }
```

Now, the model parameters are introduced along with a Boolean variable which controls whether the approximation (36) is to be used:

```
tildeT<-100; n<-6; q<-0.2; C<-100; c<- 5 ; app<-FALSE # set parameters
```

Initialize now the containers for storage of the value functions and of the continuation region:

```
t<-tildeT; VE<-rep(0, t + n +1) # initialize backward induction
cont_reg<-list(1:(tildeT+1))    #   and create containers
```

With these settings, the backward induction is performed

```
while (TRUE)   # follow backward induction
{
reward<-make_reward(t=t, tildeT=tildeT, n=n, q=q, C=C, c=c) # construct reward
V<-apply(FUN=max, cbind(reward, VE), MARGIN=1) # determine maximum of
# expected value function and reward
cont_reg[[t+1]]<- VE>reward
Tr<-make_tr_matrix(q=q, d=length(V)+1) # calculate conditional
VE<-Tr%*%c(V, C)                                  # expectation
VE<-VE[-length(VE)]
if (t>0) t<-t-1 else   break ; print(t) }
```

Thereafter, the data defining the continuation and the stopping region are extracted

```
stopping_points<-list(1:(tildeT+1))
continuation_points<-list(1:(tildeT+1))
arguments<-0:tildeT
for (t in 0:tildeT) {
stopping_points[[t+1]]<-(t- 0:(t+n))[!cont_reg[[t+1]] ]
continuation_points[[t+1]]<-(t- 0:(t+n))[cont_reg[[t+1]] ] }
```

and the regions are plotted

```
plot(x=0, y=0,  xlim=c(0, tildeT),
ylim=c(-n, tildeT), type="n", xlab="time", ylab="block difference")
for (t in arguments) {
points(x=rep(t, length(stopping_points[[t+1]])),
y=stopping_points[[t+1]], type = "l", col="red")
points(x=rep(t, length(continuation_points[[t+1]])),
y=continuation_points[[t+1]], type = "l", col="black", lty="dashed")  }
```

The result of this calculation is illustrated in Figure 2 which depicts the continuation and the stopping regions by dashed lines (in black) and by solid lines (in red) respectively. Recall that we agreed to consider the states visited by $(Z_t)_{t=0}^{T}$ for block difference not greater than n due to (19). Hence, the graph of the relevant states forms a triangle-type figure whose bottom range turns out to be the stopping region. In fact, we observe that the conditions of launching a double-spending attack are achieved if the block difference between the official and the secret branches attains some critical value (-2 in this calculation). In line with our intuition, this means that the attacker must wait until the private chain overtakes the official by at least two blocks. Thereafter, the payment shall be placed (attack launched) while the secret mining must continue until the end of the time horizon. Surprisingly, this critical value (-2) does not depend on time, thus the optimal exercise strategy is time-homogeneous, which is rarely seen in finite-horizon optimal stopping problems. This phenomenon is observed for diverse sets of parameters in all numerical calculations and can be explained by a weak dependence of the rewards on time and by a time-homogeneity at the last time point $\tilde{T} = T - n$ at which the attack can be launched having in mind that the race effectively continues until T, by construction.

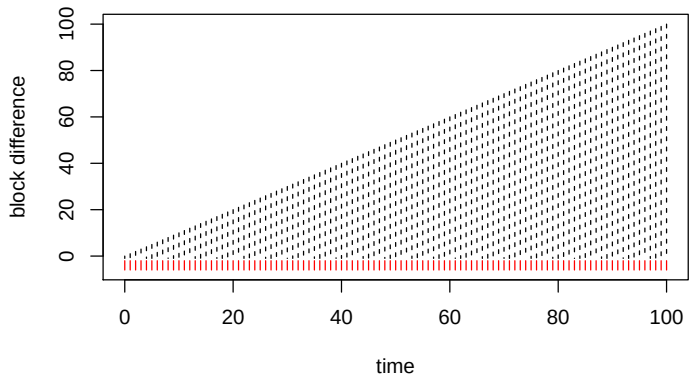

Figure 2. Solution to optimal stopping problem (18) . The continuation region is depicted by dashed lines, while and the stopping regions by solid lines.

Remark 8. *As expected, the optimal stopping policy heavily depends on model parameters and changes with number of required conformation blocks, costs, rewards and capacity ratio. The interested reader is encouraged to experiment with our code for different parameters to investigate diverse situations.*

8.2. Numerical Illustration of Optimal Switching

We illustrate the algorithm presented above by an implementation in R. First let us define the matrix for calculation (47) of conditional expectation by the same code as in the second listing from Section 8.1.

and introduce a routine for generation of switching matrix (39):

```
make_switch_matrix<-function(n) {
mat<-matrix(data=0, nrow=2+n, ncol=3)
mat[1,]<-c(1,1,1)
mat[2,]<-c(1,2,3)
for (i in 3:(nrow(mat)-1))  {mat[i,]<-c(1, i+1, i+1)}
mat[nrow(mat),]<-c(1, n+2, n+2)
return(mat) }
```

Next define functions for (42) and (43)–(45)

```
asympt_valuefct<-function(t, p, z) {
stopifnot((1<=p)&(p<=n+2)&(z>T))
(1<p)*rho*(T+1)+(2<p)*(C*(t<=T-n+p-2)-c*(t>T-n+p-2))+(p==2)*C*(t<=T-n) }
############
reward_fct<-function(t, p, z, a){
stopifnot((1<=p)&(p<=n+2)&((a==1)|(a==2)|(a==3))&(0<=t)&(t<=T))
result<-0
if (a==1){
result<- rho*z*(z>t)*(1<p) + C*(z>t)*(p==2+n)-c*(2<p)*(p<2+n) }
if ((a==2)|(a==3)){
result<- -m*(1<p) }
return(result) }
#############
scrap_fct<-function(p, z){return(-c*(2<p)*(p<=2+n))}
```

The functions for generation of containers for rewards and scrap values are

```
make_rewards<-function(t){
result<-array(data=0, dim=c(T+1, n+2, 3))
for (z in (1:dim(result)[1]))
for (p in (1:dim(result)[2]))
for (a in (1:dim(result)[3])){
result[z, p,  a]<-reward_fct(t, p, z-1, a) }
return(result) }
###############
make_scraps<-function(){
result<-array(data=0, dim=c(T+1, n+2))
for (z in (1:dim(result)[1]))
for (p in (1:dim(result)[2]))  {
result[z,p]<-scrap_fct(p, z-1)}
return(result) }
```

The conditional expectation calculation (47) is implemented as

```
cond_exp<-function(t, valuefunc, tr_mat){
lastrow<-matrix(data=0, nrow=1, ncol=n+2)
for (p in 1:(n+2)) lastrow[p]<-asympt_valuefct(t, p, T+1)
exppvaluefunc<-tr_mat%*%rbind(valuefunc, lastrow)
return(exppvaluefunc)}
```

whereas (46) is realized by

```
maximize_action<-function(t, evalfunc,alfa) {
container<-array(data=0, dim=c(T+1, n+2,3))
rewards<-make_rewards(t)
for (p in (1:(n+2)))
for (a in (1:3))  {
container[, p, a]=rewards[,p,a]+evalfunc[,alfa[p, a]]}
result=list( apply(FUN=max, X=container, MARGIN=c(1,2)),
apply(FUN=which.max, X=container, MARGIN=c(1,2)) )
names(result)<-c("opt_val", "opt_act")
return(result)}
```

which yields beyond value function maximization also the maximizing actions.

```
bellman<-function(timepoint)
{  stopifnot((0<=timepoint)&(timepoint<=T))
tr_mat<-make_tr_matrix(q=q, d=T+2)
alfa<-make_switch_matrix(n)
valuefunc<-make_scraps()
evalfunc<-valuefunc
t=T
################################
while (t>=timepoint)
{
res<-maximize_action(t, evalfunc, alfa)
evalfunc<-cond_exp(t,res$opt_val, tr_mat)
t<-t-1
}
################################
return(res)
}
```

As an illustration, define the model parameters and run strategy optimization:

```
T<-100; n<-6; q<-0.1; C<-100; c<- 5; rho<-0.1; m=0.07
###################################
res<-bellman(0)
res$opt_act[1:5,]
round(res$opt_val[1:5,], digits=3)
res$opt_val[1,2]
```

which returns

```
res$opt_act[1:5,]
[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    1    2    2    2    2    2    2
[2,]    1    1    2    2    2    2    2    1
[3,]    1    1    2    2    2    2    2    1
[4,]    1    3    2    2    2    2    2    1
[5,]    1    3    2    2    2    2    2    1
round(res$opt_val[1:5,], digits=3)
[,1]  [,2]   [,3]   [,4]    [,5]    [,6]    [,7]    [,8]
[1,]    0 0.000 -0.334 -0.234  -0.073   0.264   1.068   1.068
[2,]    0 0.100 -0.255  0.011   0.706   2.878  10.913 100.100
[3,]    0 0.200  0.182  1.395   5.308  19.787 100.141 100.200
[4,]    0 0.456  2.343  8.219  27.786 100.182 100.241 100.300
[5,]    0 3.551 11.492 34.996 100.223 100.282 100.341 100.400
res$opt_val[1,2]
[1] 0
```

Note in the last line that the result $\nu_0^*(0,2) = 0$ is as in (41), ensuring that the block chain is resilient to the double-spending attacks. When examining the strategy, we note that the optimal action is to quit (line 3, column $[,2]$) right at the beginning, even if the block difference was one and two (line 3 and 5, column $[,2]$). However, for a block difference greater than two, the double-spending attack shall be launched (line 6 and 7, column $[,2]$).

However, for a different set of parameters we obtain another interesting situation:

```
1  T<-100; n<-6; q<-0.18; C<-10; c<- 1; rho<-0.25; m=0.01
2  ################################################################################
3  res<-bellman(0)
4  res$opt_act[1:5,]
5  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
6  [1,]    1    2    2    2    2    2    2    2
7  [2,]    1    1    2    2    2    2    2    1
8  [3,]    1    2    2    2    2    2    2    2
9  [4,]    1    2    2    2    2    2    2    2
10 [5,]    1    3    2    2    2    2    2    2
11 round(res$opt_val[1:5,], digits=3)
12 [,1]  [,2]  [,3]  [,4]   [,5]   [,6]   [,7]   [,8]
13 [1,]    0 0.012 0.001 0.049  0.127  0.258  0.474  0.474
14 [2,]    0 0.250 0.119 0.267  0.542  1.087  2.291 10.250
15 [3,]    0 0.567 0.473 0.934  1.850  3.871 10.557 10.557
16 [4,]    0 0.916 1.426 2.707  5.251 10.865 10.865 10.865
17 [5,]    0 2.006 3.616 6.461 11.172 11.172 11.172 11.172
18 > res$opt_val[1,2]
19 [1] 0.01182954
```

Here we observe that the attack is economically justifiable (line 19). Having started chain bifurcation at block difference zero, the secret mining is pursued (line 6, column $[,2]$). However, being one block ahead, it is better to publish the secret chain to pocket the reward for a single block (line 7, column $[,2]$). If the block difference is two or three, then one shall continue secret mining (lines 8, 9, column $[,2]$), but for larger block difference, an attack (line 10, column $[,2]$) should be launched. If the attack is already launched, one shall continue secret mining, with the exception when all confirming blocks are received and the attacker's advantage is exactly one block (line 7, column $[,8]$), in which case the secret branch should be published.

Remark 9. *As illustrated above, the optimal double-spending strategy exhibits a relatively complex behavior. Due to its time-changing nature and non-obvious parameter dependence, it is difficult to face all quantitative aspects by diagrams and numerical case studies. For practical use, we suggest that a detailed investigation should be performed on a given parameter set using our code.*

Finally, let us numerically investigate the values $\nu_0^*(0,2)$ for different gains $C = 100, 50, 30$ from a potentially successful double spending depending on the proportion $q \in [0.15, 0.3]$ between the mining capacity of the attacker and the total mining power.

```
T<-100; n<-6; q<-0.1; C<-100; c<- 5; rho<-0.1; m=0.07
ratio<-seq(from=0.2, to=0.3, length=20)
values<-seq(from=0, to=0, length=length(ratio))
################################
for (i in 1:length(ratio)) {q<-ratio[i]; values[i]<-bellman(0)}
plot(ratio, values, type="l", col="blue", xlab="mining capacity ratio",
ylab="value function", lty=1, lwd=2)
################################
C<-50
for (i in 1:length(ratio)) {q<-ratio[i]; values[i]<-bellman(0)}
points(ratio, values, type="l", col="red", lty=2, lwd=2)
################################
C<-30
for (i in 1:length(ratio)) {q<-ratio[i]; values[i]<-bellman(0)}
points(ratio, values, type="l", col="magenta", lty=3, lwd=2)
```

This code generates graphs depicted in Figure 3. In line with our intuition, the curves confirm that the profitability of an optimal double-spending strategy increases with the potential gain C and the proportion q of attacker's mining capacity. From this picture, we also infer the minimal capacity ratio required for a successful double spending. For instance, the lowest curve shows that with $C = 30$ this minimal ratio is around 24%. However, this value decreases to approximately 22% and further to

20% for $C = 50$ and $C = 100$. Let us emphasize that such calculations can be used to determine the block number required to secure a transaction depending on its size.

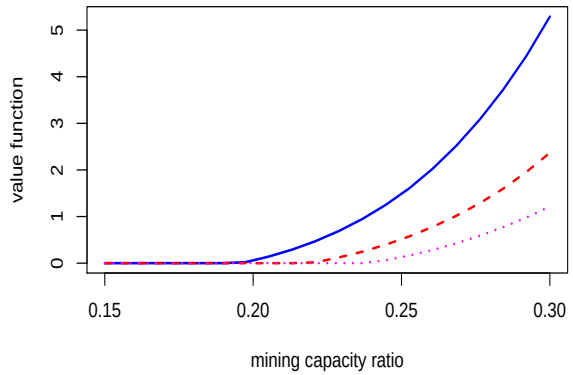

Figure 3. The value $\nu_0^*(0,2)$ of the double-spending attack for $n = 6$ confirming blocks depending on the mining ratio $q \in [0.15, 0.3]$ calculated for $C = 100$, $C = 50$, and $C = 30$ (solid, dashed, and dotted line respectively).

9. Conclusions

Unfortunately, even relatively unsophisticated (classical textbook-style) double-spending attacks happen regularly. These malicious actions cause huge losses to investors and jeopardize the perspectives of the promising blockchain technology. In fact, the situation is worrying. The point here is that the double-spending problem actually concerns more than a single payment which may disappear later. The sheer possibility to rewrite the ledger with a deep re-organization of its blocks may cause enormous consequences.

In this work, we show that planning an attack on a PoW-based system can be formulated as optimal sequential decision problem. Therefore, we consider two cases: A simplified model of a double-spending attack, which can be treated as an optimal stopping problem, and a more detailed modeling which requires an optimal stochastic switching toolbox.

In the optimal stopping situation, the strategy consists of a secret mining, followed by a later payment. The optimal payment moment is determined by the length difference between the official and secret chains since their fork-off and depends on model parameters (mining capacity ratio, confirming block number and on the revenue/loss from the success/failure of the attack). A more complex stochastic switching model upgrades this framework by introducing the option to abandon secret mining at any time. Furthermore, a switching model provides also a more realistic context since it takes into account mining costs and rewards for published blocks. Most importantly, the optimal strategy can be used to determine whether it is worth attacking the PoW-based system. This insight may allow important conclusions on its vulnerability. However, to address this topic within an entirely realistic situation, the present models must be further developed to include propagation delay, and uncertainty in observations. Furthermore, complex consensus protocols based on (delegated) proof of stake, proof of storage, proof of burn or their combinations must be investigated from a similar perspective. Finally, also the possibility to slow down the honest mining by diverse malicious actions (in the spirit of [14]) must be examined. Here, a deeper understanding of the natural bifurcations of the official chain (which slows down its growth) and the attacker's opportunity to enforce it (by publishing blocks, jamming the network and causing propagation delays) are crucial. All these problems must be systematically addressed to improve stability of block chain systems.

Acknowledgments: This work would not have been possible without the advice, help, kind support, and very significant contributions of Peter Taylor. The author would also like to thank to anonymous referees for their criticism and remarks which helped us improving this work. In particular, the author expresses deepest gratitude to the referee suggesting an investigation of PoW stability in terms of optimal switching techniques. Furthermore, the author appreciates helpful communication with the editor of MDPI and thanks F. Hinz and P. Hinz for discussions and Vonida UG (haftungsbeschränkt) for their support.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Let us derive the assertions (6) and (7) from the relation between mining capacities.

Proof. The time, required to complete the next block follows an exponential distribution since the process of mining can be described by repeated attempts to solve a cryptographically puzzle by independent random trials. Indeed, taking into account that the waiting time to first success in a sequence of Bernoulli trials is geometrically distributed and the time spent on each trial is short, the exponential distribution provides an excellent approximation for the time required to complete a next block. For this reason, the block numbers mined in the secret and official branch since their fork-off can be described in physical time $u \in \mathbb{R}_+$ by independent Poisson processes $(N^S_u)_{u\in\mathbb{R}_+}$, $(N^O_u)_{u\in\mathbb{R}_+}$. Furthermore, the corresponding intensities $\lambda^S, \lambda^O \in]0,\infty[$ are determined by the mining capacity ratios and the total difficulty (for details, see [1,5]) and are proportional $\lambda^S = \lambda q$, $\lambda^O = \lambda(1-q)$ to the capacity fractions $q, (1-q) \in]0,1[$ of the miners. Therefore, the factor $\lambda \in]0,\infty[$ incorporates the difficulty of mining. That is, the probability of having mined $j \in \mathbb{N}$ secret blocks during a time required for the completion of one official block is given by (the author thanks Florian Hinz)

$$\begin{aligned}
&\mathbb{P}(N^S_t = j | t \text{ is the first jump time of } (N^O_u)_{u\in\mathbb{R}_+}) = \\
&= \int_0^\infty \frac{(\lambda^S t)^j}{j!} e^{-t\lambda^S} e^{-t\lambda^O} \lambda^O dt \\
&= \lambda^O (\lambda^S)^j \int_0^\infty \frac{t^j}{j!} e^{-t(\lambda^S+\lambda^O)} dt \\
&= \lambda^O (\lambda^S)^j \frac{1}{(\lambda^S+\lambda^O)^{j+1}} = \frac{\lambda^O}{\lambda^S+\lambda^O} \left(\frac{\lambda^S}{\lambda^S+\lambda^O}\right)^j = (1-q)q^j.
\end{aligned}$$

□

The proof of Lemma 1 is given below:

Proof.

(a) To show (32), recall that given $x \in \mathbb{Z}$, $n \in \mathbb{N}$ with $x \geq -n$ and $x' \in \{1,\dots,x+n+1\}$ the probability $\mathbb{P}(Z_{n+1} = x' \mid Z_0 = x)$ is the sum over probabilities of

$$\begin{gathered}\text{all trajectories of } (Z_i)_{i=0}^{n+1} \text{ which} \\ \text{start at } x \text{ and finish at } x'.\end{gathered} \tag{A1}$$

Please note that each such trajectory cannot exceed $x+n+1$ since at each time i, the Markov chain can jump up $Z_{i+1} = Z_i + 1$ by one unit at most. For the same reason, each trajectory (A1) also cannot go below $-n$ since otherwise it would not reach $x' \in \{1 \dots x+n+1\}$, i.e., the dynamics $(Z_i)_{i=0}^{n+1}$ can be equivalently replaced by $(Z_i^{(l,u)})_{i=0}^{n+1}$ in (A1) if (l,u) satisfies $l \leq -n$ and $x+n+1 \leq u$. The transition probables of $(Z_i^{(l,u)})_{i=0}^{n+1}$ over $n+1$ steps can be obtained from

the entries of the power $p^{(l,u)\ n+1}$ of its transition matrix $p^{(l,u)}$, which shows (32) . Next, in order to determine (31), we use (32) to observe that

$$\begin{aligned}\mathbb{P}(Z_{n+1} \leq 0 \,|\, Z_0 = x) &= 1 - \sum_{x'=1}^{x+n+1} \mathbb{P}(Z_{n+1} = x' \,|\, Z_0 = x) \\ &= 1 - \sum_{x'=1}^{x+n+1} (p^{(l,u)})^{n+1}_{x,x'} = \sum_{y=l-1}^{0} (p^{(l,u)})^{n+1}_{x,y}.\end{aligned}$$

(b) Consider $\mathbb{P}(\min_{i=0}^{k} Z_i \leq 0 \,|\, Z_0 = x')$ as the sum over probabilities of all sample paths of $(Z_i)_{i=0}^{k}$ which start at $x' \geq 1$ and enter the set of non-positive states. With the same arguments as in the proof of a), the process $(Z_i)_{i=0}^{k}$ can replaced in (A1) by a truncated dynamic $(Z_i^{(l,u)})_{i=0}^{k}$ with (l,u) satisfying $l = 1$ and $x' + k \leq u$, which gives

$$\mathbb{P}(\min_{i=0}^{k} Z_i \leq 0 \,|\, Z_0 = x') = \mathbb{P}(\min_{i=0}^{k} Z_i^{(l,u)} \leq 0 \,|\, Z_0^{(l,u)} = x')$$

and finishes the proof.

□

References

1. Antonopoulos, A.M. *Mastering Bitcoin: Programming the Open Blockchain*, 2nd ed; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2017.
2. Nakomoto, S. A Peer-to-Peer Electronic Cash System. Available online: https://git.dhimmel.com/bitcoin-whitepaper/ (accessed on 16 January 2020).
3. Rosenfeld, M. Analysis of Hashrate-Based Double Spending. *arXiv* **2014**, arXiv:1402.20092014.
4. Gruenspan, C.; Perez-Marco, R. Double Spend Races. *arXiv* **2017**, arXiv:1702.02867.
5. Goebel, J.; Keeler, H.P.; Krzesinski, A.E.; Taylor, P.G. Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. *Perform. Eval.* **2016**, *104*, 23–41. [CrossRef]
6. Gervais, A.; Karame, G.O.; Wüst, K.; Glykantzis, V.; Ritzdo, H.; Capkun, S. On the security and performance of proof of work blockchains. In Proceedings of the 2016 ACM SIGSAC, Vienna, Austria, 24–28 October 2016; pp. 3–16.
7. Sapirshtein, A.; Sompolinsky, Y.; Zohar, A. Optimal selfish mining strategies in bitcoin. In Proceedings of the International Conference on Financial Cryptography and Data Security, Christ Church, Barbados, 22–26 February 2016; pp. 515–532.
8. Sompolinsky, Y.; Zohar, A. Bitcoin's security model revisited. *arXiv* **2016**, arXiv:1605.09193.
9. Nguyen, G.-T.; Kim, K. A survey about consensus algorithms used in blockchain. *J. Inf. Process. Syst.* **2018**, *14*, 101–128.
10. A probabilistic analysis of the Nxt forging algorithm. A survey about consensus algorithms used in blockchain. *Ledger* **2016**, *1*, 69–83.
11. Li, Q.-L.; Ma, J.-Y.; Chang, Y.-X.; Ma, F.-Q.; Yu, H.-B. Markov processes in blockchain systems. *Comput. Soc. Netw.* **2019**, *6*, 5. [CrossRef]
12. Kawase, Y.; Kasahara, S. Transaction-confirmation time for bitcoin: A queueing analytical approach to blockchain mechanism. In Proceeding of the QTNA, Qinhuangdao, China, 21–23 August 2017.
13. Chang, Y.-X.; Li, Q.-L.; Ma, J.Y. A survey about consensus algorithms used in blockchain. *Lect. Notes Comput. Sci.* **2018**, *11*, 25–40.
14. Eyal, I.; Sirer, E.G. Majority is not enough: bitcoin mining is vulnerable. *Commun. ACM* **2018**, *61*. [CrossRef]
15. Miller, A.; Shi, E.; Nayak, K.; Kumar, S. Stubborn mining: generalizing selfish mining and combining with an eclipse attack. In Proceedings of the IEEE European Symposium on Security and Privacy, Saarbrucken, Germany, 21–24 March 2016; pp. 305–320.
16. Goffard, P.-O. Fraud Risk Assessment Within Blockchain Transactions. 2019, in press.
17. Jang, J.; Lee, H. Profitable double-spending attacks. *arXiv* **2019**, arXiv:1903.01711.

18. Bäuerle, N.; Rieder, U. *Markov Decision Processes with Applications to Finance*; Springer: Heidelberg, Germany, 2011.
19. Pham, H. *Continuous-Time Stochastic Control and Optimization With Financial Applications*; Springer Science & Business Media: Berlin, Germany, 2009; Volume 61.
20. Powell, W.B. *Approximate Dynamic Programming: Solving the Curses Of Dimensionality*; Wiley: Hoboken, NJ, USA, 2007.
21. Hinz, J.; Tarnopolskaya, T.; Yee, J. Efficient algorithms of pathwise dynamic programming for decision optimization in mining operations. *Ann. Oper. Res.* **2018**, *6*, 1–33. [CrossRef]
22. Hinz, J. Optimal stochastic switching under convexity assumptions. *SIAM J. Control Optim.* **2014**, *52*, 164–188. [CrossRef]
23. Hinz, J.; Yap, N. Algorithms for optimal control of stochastic switching systems. *Theory Probab. Appl.* **2015**, *60*, 770–800. [CrossRef]
24. Hinz, J.; Yee, J. Optimal forward trading and battery control under renewable electricity generation. *J. Bank. Finance* **2018**, *95*, 244–254. [CrossRef]
25. Hinz, J.; Yee, J. Stochastic switching for partially observable dynamics and optimal asset allocation. *Int. J. Control* **2017**, *90*, 553–565. [CrossRef]
26. Hinz, J.; Yee, J. rcss: R package for optimal convex stochastic switching. *R J.* **2018**, *10*, 38–54. [CrossRef]