HEX Financial Audit

Financial Analysis of Stakes

October 2019

By Coinfabrik

Contents

Summary

HEX is an ERC-20 token and fully-automated contract deployed on the Ethereum network used to recreate a traditional banking product called "Time Deposit".

CoinFabrik was asked to audit the contracts for the HEX project. In particular, we were asked to verify that longer stakes pay better than shorter re-compounding stakes. This document discusses the issue and provides an insight into why longer stakes are better than short re-compounding stakes when using the same resources. The audited commit is 640906556dd14b2b57902185557b36f8d4251806.

We were able to verify that longer stakes pay better than short re-compunding stakes.

We show how this happens for certain (most) protocol parameters. Further, when different users are staking the same HEX at the same time, then we show that longer stakes always return more HEX than composing shorter stakes. We expose one edge case of little practical importance: if only one person ever stakes during a time window, then that person might be able to get bigger profits with short-recompounding stakes than with a longer stake. It is certain that this will never be the case. Our descrition describes these conditions and the case in detail.

Introduction

The Hex protocol allows a user, with a valid address, to start a stake at any time, for an amount of hearts and a number of (staked) days. At any later moment, the user may end the stake. If certain conditions are satisfied, the end stake operation is allowed, and the user's address is minted with the **stake return**.

A stake may be started with a call to the `stakeStart()` function, with an input that includes the stake h (an integer representing the number of hearts invested) and the stake period d (an integer representing the number of staked days). When the stake is started, h hearts are burned from the user's address and the **locked day** is recorded as the following day (if we are beyond the 'Claim Phase start day', else it is computed differently).

The stake may be ended with the user calling the `stakeEnd()` function. After checking some conditions, the contract computes the stake return and mints the user's address with

this amount.

The formula computing this payout is complex and depends on protocol specification and what other protocol parties are doing. In particular, if the number of served days (the days elapsed since the locked day until the day `stakeEnd` is called) is as big as the staked days, then the stake return is computed as the original stake plus a **payout** (else, if he ends the stake too early or too late, he may incur in some penalties).

In order to compare long and short stakes, we first note that it suffices to show that this holds for one long stake versus two shorter stakes. A proof of this statement follows in the appendix.

It thus suffices to compare following two strategies.

Strategy 1 (long stake): A user stakes h hearts for d days, where d, h are positive integers, and obtains a stake return of R_1 hearts.

Strategy 2 (short recompounding stake): A user stakes h hearts for d_1 days, obtains a stake return which he re-stakes for d_2 days, after which he ends the stake to receive a stake return of R_2 hearts. We assume that d_1, d_2 are positive integers that verify $d_1 + d_2 \leq d - 1$.

Strategy 1:	Strategy 2:
Call $\texttt{st} := \texttt{stakeStart}(h, d)$ Wait day $d + 1$ days Call $\texttt{stakeEnd}(\texttt{st})$ Get minted R_1 hearts	Call $\texttt{st}_1 := \texttt{stakeStart}(h, d_1)$ Wait $d_1 + 1$ days Call $\texttt{stakeEnd}(\texttt{st}_1)$ Get minted Q hearts Call $\texttt{st}_2 := \texttt{stakeStart}(Q, d_2)$ Wait $d_2 + 1$ days Call $\texttt{stakeEnd}(\texttt{st}_2)$ Get minted R_2 hearts

Since when a user calls `stakeStart` on a day i the locked day is actually $i + 1$ ($\texttt{newLockedDay} = \text{g.\_}\texttt{currentDay} + 1$), it follows that the sum of the staking periods for Strategy 2 is one day shorter than the staking period for Strategy 1. That is why we ask that $d_1 + d_2 \leq d - 1$ so that both strategies span at most d days.

Protocol Description

In what follows we first produce a formalization that helps us to recreate the functions that compute the payout in order to conclude that longer pays better.

Let us denote by $F(d;h)$ the payout received by a user after calling $\texttt{stakeStart}(h,d)$ with $\texttt{newStakedHeart} : h$ hearts and $\texttt{newStakedDays} : d$ days, and $d+1$ days later calling $\texttt{stakeEnd}()$. (So the stake return is $h + F(h,d)$.) According to the code, it follows that

$$F(h;d) := \sum_{i=i_0+1}^{d} \frac{\texttt{dailyData}[i].\texttt{dayPayoutTotal} \cdot \texttt{stakeShares}(h,d,i)}{\texttt{dailyData}[i].\texttt{dayStakeSharesTotal}} \tag{1}$$

(We denote the day the stake is started by i_0 to make notation simpler.) We have removed bonuses from penalties from this calculation.

Here dailyData is a (global) array, that has one element for each elapsed day. These elements are 3 named values. More generally, g denotes the global cache, a variable that stores some global objects which we will continue to introduce in this section. For everyday elapsed day, before any start/end stake operation, the function $\texttt{storeDailyDataBefore}()$ is called and the array dailyData is updated to cover all days from the Claim Phase Start Day until the current day. The following formulas are used:

$$\texttt{dailyData}[i].\texttt{dayPayoutTotal} = (\texttt{TOTAL\_SUPPLY}+\texttt{g.\_lockedHeartsTotal})\cdot\texttt{INFLATION} \tag{2}$$

and

$$\texttt{dailyData}[i].\texttt{dayStakeSharesTotal} = \texttt{uint72}(\texttt{g.\_stakeSharesTotal}) \tag{3}$$

where TOTAL_SUPPLY and INFLATION are constants, and g._lockedHeartsTotal and g._stakeSharesTotal are global variables.

Every time $\texttt{stakeStart}(H,D)$, for some integers H and D, H hearts are added to the global variable g._lockedHeartsTotal, and whenever this stake is ended, H hearts are deducted from g._lockedHeartsTotal. No other operation modifies this variable.

We follow to discuss the denominator of (1). Let s_0 denote the value held by global variable g._stakeSharesTotal on day $i = 0$. The variable is modified by two events:

1. Before (3) is evaluated, the following code runs:

$$\begin{aligned} &\texttt{g.\_stakeSharesTotal} += \texttt{g.\_nextStakeSharesTotal} \\ &\texttt{g.\_nextStakeSharesTotal} = 0 \end{aligned} \tag{4}$$

   In turn, other than in (4), g._nextStakeSharesTotal is only modified whenever a stake is started. At that moment, the contract computes

$$\texttt{g.\_nextStakeSharesTotal} += \texttt{newStakeShares}. \tag{5}$$

   The variable newStakeShares is computed according to (7) that we describe later.

2. Also, when a stake is ended, the function stakeEnd() calls the function _stakeUnlock() which in turn executes

$$\texttt{g.\_stakeSharesTotal} -= \texttt{st.\_stakeShares}. \tag{6}$$

   Here st denotes the stake object, and st._stakeShares, which denotes the shares for this stake, is equal to newStakeShares above (5).

Hence, when the stake starts, newStakeShares hearts are added to g._stakeSharesTotal; and the same amount is deducted later when the stake ends.

The value newStakeShares is a stake parameter. It is computed when the stake is started according to (7) below. Let i_0 denote the day the stake is started and say it is started with the parameters d, h. Then, for the older version of the code, we have that

$$\texttt{newStakeShares} = \left(h + h\frac{(d-1)\cdot \texttt{BPB} + h\cdot \texttt{LPB}}{\texttt{LPB}\cdot\texttt{BPB}}\right)\cdot\frac{\texttt{SH}}{\texttt{sh}[i]} \tag{7}$$

where LPB and BPB are protocol constants, SH is short hand for the protocol constant SHARE_RATE_SCALE and $\texttt{sh}[i]$ denotes the value held by the global variable g._shareRate on the ith day.

The global variable g._shareRate is initialized as SH and is only updated when a stake is ended. When a stake ends, function _shareRateUpdate is called. It takes as input the global and stake objects and the stake return (payout plus original stake) and returns, for a stake locked on day i_0 with a return of $r = h+p$, the value $\max\{\texttt{sh}[i-1], \texttt{newShareRate}\}$ where

$$\begin{aligned} \texttt{newShareRate} &= (\texttt{stakeReturn} + \texttt{bonusHearts})\cdot\frac{\texttt{SH}}{\texttt{stakeShares}(h,d,i_0)} \\ &= \left(r + r\frac{(d-1)\cdot\texttt{BPB}+r\cdot\texttt{LPB}}{\texttt{LPB}\cdot\texttt{BPB}}\right)\cdot\frac{\texttt{SH}}{\texttt{stakeShares}(h,d,i_0)} \\ &= \left(r + r\frac{(d-1)\cdot\texttt{BPB}+r\cdot\texttt{LPB}}{\texttt{LPB}\cdot\texttt{BPB}}\right)\cdot\frac{\texttt{SH}}{\left(h+h\frac{(d-1)\cdot\texttt{BPB}+h\cdot\texttt{LPB}}{\texttt{LPB}\cdot\texttt{BPB}}\right)\cdot\frac{\texttt{SH}}{\texttt{sh}[i_0]}} \end{aligned} \tag{8}$$

We can now reconstruct formula (1) for the payout of a stake $F(h,d)$.

$$\begin{aligned} F(h;d) &= \sum_{i=i_0+1}^{i_0+d} \frac{\texttt{dailyData}[i].\texttt{dayPayoutTotal} \cdot \texttt{stakeShares}(h,d,i)}{\texttt{dailyData}[i].\texttt{dayStakeSharesTotal}} \\ &= \sum_{i=i_0+1}^{i_0+d} \frac{(\texttt{TOTAL\_SUPPLY} + \texttt{g.lockedHearts}) \cdot \texttt{INFLATION} \cdot \texttt{stakeShares}(h,d,i)}{\sum_{st \in D_i} \texttt{stakeShares}(h_{\texttt{st}}, d_{\texttt{st}}, i_{\texttt{st}})} \end{aligned} \tag{9}$$

where D_i is the set of all the stakes $\texttt{st}$ that were started on day $i_{\texttt{st}}$ (for $d_{\texttt{st}}$ days and with a stake of $h_{\texttt{st}}$ hearts) and not ended by day i; and

$$\texttt{stakeShares}(h_{\texttt{st}}, d_{\texttt{st}}, i_{\texttt{st}}) = \left(h_{\texttt{st}} + h_{\texttt{st}} \frac{(d_{\texttt{st}} - 1)\texttt{BPB} + h_{\texttt{st}} \cdot \texttt{LPB}}{\texttt{BPB} \cdot \texttt{LPB}} \right) \cdot \frac{\texttt{SH}}{\texttt{sh}[i_{\texttt{st}}]}$$

A quick glance at formula (9) shows that its value is affected by the initial values $s_0, \texttt{sh}[i_0]$, the number of hearts h staked and the number of staked days d, as well as the stakes that are started or ended between the start and end of the stake under analysis. This creates a complex situation in which making statements about our inequality seem far from reachable. In the next section, we consider a simplification of the model which allows us to make some statements in the way of proving that longer pays better.

Simplified analysis

We consider a slightly simplified model in which we shall compare strategies 1 and 2. We assume that no (other) stakes are started or ended between the day the stake is started (day i_0) and the day it is ended (day $i_0 + d + 1$). Further, we make another assumption to simplify how stake returns are computed. All these conditions are detailed below. We call these assumptions 'conditions (C)'.

1. When analysing Strategy 1 or 2 let i_0 be the day the first stake is started and let $i_0 + d + 1$ denote the date the stake is ended. Then, no other stakes are either started or ended between day i_0 and day $i_0 + d + 1$.

2. We ignore the term

$$\texttt{payout} += \texttt{waasRound} + \texttt{\_calcAdoptionBonus}(\texttt{g}, \texttt{waasRound});$$

that gets added to the payout when the day the stake is started happens before CLAIM_PHASE_END_DAY and the stake is ended after that day. CLAIM_PHASE_END_DAY is a protocol constant.

3. We have further removed from the analysis –again for the sake of simplicity– penalties inflicted upon other users that are rewarded to stakers.

The HEX protocol establishes some caps for the number of staked days and the number of staked hearts. When these caps are exceeded the math changes. Caps are not exceeded if we only start stakes for less than $D < 3641$ days (approximately 10 years) and $H < 15e15$ hearts.

Also, we define the following parameters that are used through the description:

- TOTAL_SUPPLY is a constant; it is defined outside the protocol.
- We write allocSupply$[i]$ for the value of TOTAL_SUPPLY$+$g._lockedHeart on day i and note that we can safely assume that g._lockedHeart changes daily in our simplified model.
- BPB $= 15e16$ is a constant,
- LPB $= 1820$ is a constant,
- SH $:=$ SHARE_RATE_SCALE $= 100000$ is a constant,
- INFLATION $= 10000/100448995$ is a constant,
- g.shareRate is a global variable that is updated immediately after a stake is ended. However (again) by our hypotheses (C) we can assume that it only changes daily and denote its value on the ith day by sh$[i]$.
- The auxiliary function stakeShares(H, D, i) computes stake shares on the i-th day for a stake of H hearts and D days:

$$\texttt{stakeShares}(H, D, i) = \left(H + H \frac{(D-1) \cdot \texttt{BPB} + H \cdot \texttt{LPB}}{\texttt{LPB} \cdot \texttt{BPB}} \right) \cdot \frac{\texttt{SH}}{\texttt{sh}[i]}$$

- g._stakeSharesTotal is a global variable. Let us denote by s_0 the value of g._stakeSharesTotal on day 0.

Finally, without loss of generality we can assume that $i_0 = 0$ since the value of this has no effect on results.

If conditions (C) hold, then for all h, d such that $h < 15e15$ and $d < 3641$ it holds that:

$$F(h;d) := d\frac{\texttt{allocSupply}[1] \cdot \texttt{INFLATION} \cdot \left(h + h\frac{(d-1)\cdot\texttt{BPB}+h\cdot\texttt{LPB}}{\texttt{LPB}\cdot\texttt{BPB}}\right)\frac{\texttt{SH}}{\texttt{sh}[0]}}{s_0 + \left(h + h\frac{(d-1)\cdot\texttt{BPB}+h\cdot\texttt{LPB}}{\texttt{LPB}\cdot\texttt{BPB}}\right)\frac{\texttt{SH}}{\texttt{sh}[0]}}. \tag{10}$$

Here, $\texttt{allocSupply}[1] = \texttt{allocSupply}[0] + h$, $\texttt{sh}[0]$ is the value held by g._shareRate on the day $i_0 = 0$, and s_0 is the sum of all the stake shares started and not ended before day $i_0 = 0$.

Moreover, if d_1, d_2 are integers such that $d_i < 1820$ (for $i \in \{1, 2\}$), then $F(h; d_1, d_2)$ can be computed from the above by

$$F(h;d_1,d_2) = F(h,d_1) + d_2\frac{\texttt{allocSupply}[d_1+1] \cdot \texttt{INFLATION} \cdot \texttt{stakeShares}(Q, d_2, d_1+1)}{s_0 + \texttt{stakeShares}(Q, d_2, d_1+1)} \tag{11}$$

where $Q = h + F(h, d_1)$, $\texttt{allocSupply}[d_1+1] = \texttt{allocSupply}[0] + Q$,

$$\texttt{stakeShares}(Q, d_2, d_1+1) = \left(Q + Q \cdot \frac{(d_2-1)\cdot\texttt{BPB} + Q\cdot\texttt{LPB}}{\texttt{LPB}\cdot\texttt{BPB}}\right) \cdot \frac{\texttt{SH}}{\texttt{sh}[d_1+1]}$$

and $\texttt{sh}[d_1+1]$ is computed by (8).

Protocol Analysis

Longer pays better

We shall show that longer is better in terms of stakes. We make use of the formulas above. Formulas for $F(h;d)$ and $F(h;d_1,d_2)$ allow us to compute payouts if condition (C) holds. The formulas depend on some parameters (other than h,d):

$$s_0, \texttt{sh}[0] \text{ and } \texttt{allocSupply}[0]$$

Hence, for every example we need to establish their values.

First, we illustrate the point with an example. Below find a graph where $s_0 = 1e7$, $\texttt{sh}[0] = 1.1e5$, $\texttt{allocSupply}[0] = 1.8e19$ and we draw two lines:

- The blue line describes stake-payouts according to the long strategy (Strategy 1) whereby h hearts are staken for $d = 700$ days. The x-axis describes the input stakes and the y-axis describes the payout.
- The red line describes stake-payouts according to a short strategy (Strategy 2) whereby h hearts are staken for $d = 350$ days, and once the stake return of$h + F(h, 350)$ is minted to the user, this is immediately re-staken for 349 days, and then the stake is ended. The x-axis describes the input stakes h and the y-axis describes the (total) payout.

It can be seen that $F(h;700) > F(h;350,349)$ in the cases depicted by the above graph (Figure 1).

To argument that there are no exceptions to this behavior, we need to show that for all possible values of the input and parameters, the inequality $F(h;d) < F(h;d_1,d_2)$ holds. One such argument would need to cover all possible values of d, d_1, d_2, h, s_0, $\texttt{sh}[0]$ and $\texttt{allocSupply}[0]$.

Notice that once d and d_1 are fixed, with $d_1 < d$, then the best possible d_2 we can choose (i.e., the one that makes $F(h;d_1,d_2)$ bigger) is $d_2 := d - d_1 - 1$, which is the biggest admissible value for d_2. Alternatively, once we fix d, we can pick δ with $0 < \delta < 1$

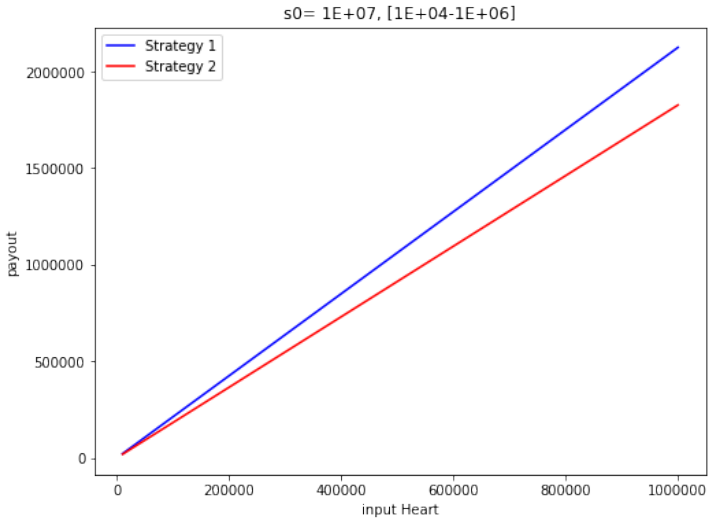

Figure 1: Simple example

and have $d_1 := \lfloor \delta(d-1) \rfloor$ and $d_2 := (d-1) - \lfloor \delta(d-1) \rfloor$[1]. So that once fixed d, moving δ in $(0,1)$ covers all the values of d_1 and d_2.

This is done next. We fix $s_0 = 10^7$, $\texttt{sh}[0] = 1.1e5$, $\texttt{PAYOUT\_TOTAL+g.\_lockedHeart} = 2.3e12$, $h = 10^8$, $d = 700$ and check what happens when δ moves in $(0,1)$ (Figure 2). Obviously, since Strategy 1 is independent of δ, the payout remains constant. The interesting point here is that Strategy 2 gets closer to Strategy 1 when δ is close to 0 or 1 and the worse possible payout is attained when $\delta = 0.5$.

The conclusion of this particular example is that the closer a short re-compounding stake gets to a long stake, the better is the return.

Extrapolating

The fact that longer pays better for a specific set of parameters in no guarantee that this happens for any other choice of parameters.

The parameters for our model are $h, d, \delta, s_0, \texttt{sh}[0]$ and $\texttt{allocSupply}[0]$. The values

[1] We use the notation $\lfloor x \rfloor := \max\{m \in \mathbb{Z} : m \leq x\}$ for the floor function that returns the biggest integer smaller than x.

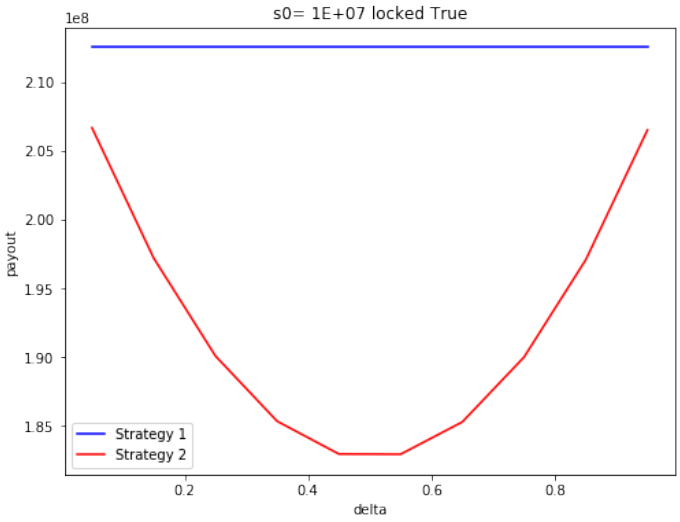

Figure 2: Moving δ

for s_0 and $\texttt{sh}[0]$ depend on the other stakes happening, and $\texttt{allocSupply}[0]$ depends on other stakes happening and the value of the constant `TOTAL_SUPPLY` (which is set outside the protocol). But, while there are some protocol limitations for h, d and δ, these can be set arbitrarily by any user and they are in control of a user which may be trying to find examples where shorter re-compounding stakes are better than longer stakes.

As a first attempt, we select different choices of h and d and draw a graph on how the payouts vary depending on δ. The figure below (Fig. 3) shows that the same situation as in Figure (2) repeats for different choices of h and d. In fact, numerical experiments with the derivative with respect to δ of the function that computes the payout for Strategy 2 have all the same form, almost a line that goes from the negative to the positive–implying that the function reaches a minimum when $\delta \sim 0.5$.

Another matter is selecting values for the parameters not controlled by the user; namely, $s_0, \texttt{sh}[0]$ and $\texttt{allocSupply}[0]$. What can we expect from these?

We arbitrarily chose $\texttt{allocSupply}[0] = 1.8e19$ as this value is set outside of the protocol. For s_0 and $\texttt{sh}[0]$ we choose two values for each parameter ($s_0 \in \{1e12, 1e16\}$ and $\texttt{sh}[0] \in \{1e12, 1e17\}$) and display the four possible graphs. Even moving the values within these ranges, has a similar behavior.

The case is that the maximum for $F(h; \lfloor \delta(d-1) \rfloor, d - 1 - \lfloor \delta(d-1) \rfloor)$ (Strategy 2) is attained when δ is 0 or 1. So the (long) Strategy 1 always wins over the (short

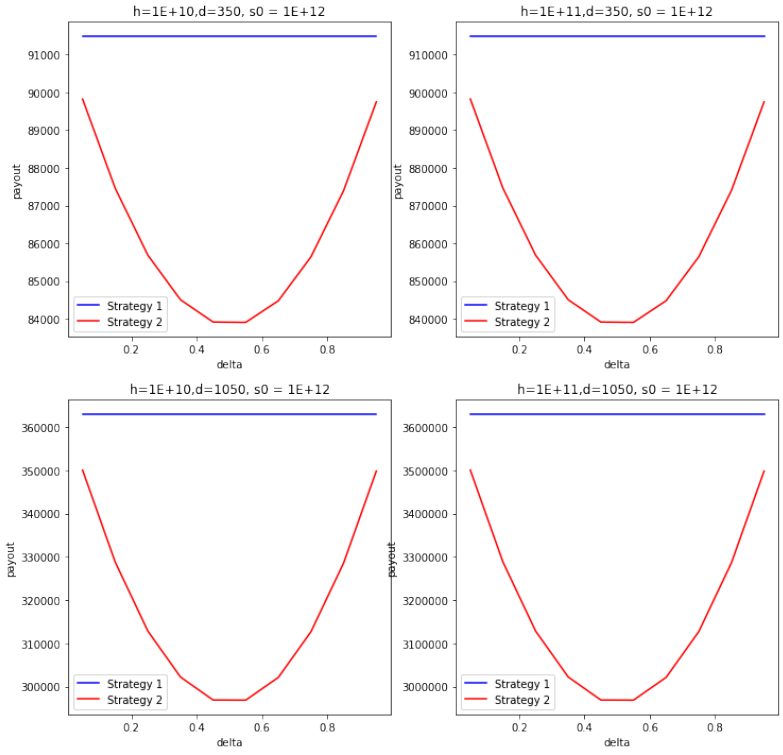

Figure 3: Moving δ for different values of h, d

re-compounding) Strategy 2 when the parameters are in the ranges we have selected.

Actually, if we now select s_0 with increasingly big values but do not update `shareRate`, then Strategy 2 gets close to Strategy 1 (but never surpasses it). Something analogous happens if we select $\texttt{sh}[0]$ to be increasingly large but keep s_0 constant (e.g., because after some time many users ended their stake, no new users started a stake and the `shareRate` grew in the interim). Yet this is probably unlikely to happen in a protocol run.

The next section analyses some anomalous cases when short re-compounding stakes are better than a long stake.

Anomalous cases

When the protocol is initialized, we have that $\texttt{shareRate} = 1e5$. Lets assume $\texttt{sh}[0] = \texttt{SHARE\_RATE\_SCALE} = 1e5$. Moreover, let us keep conditions (C) and in particular, we

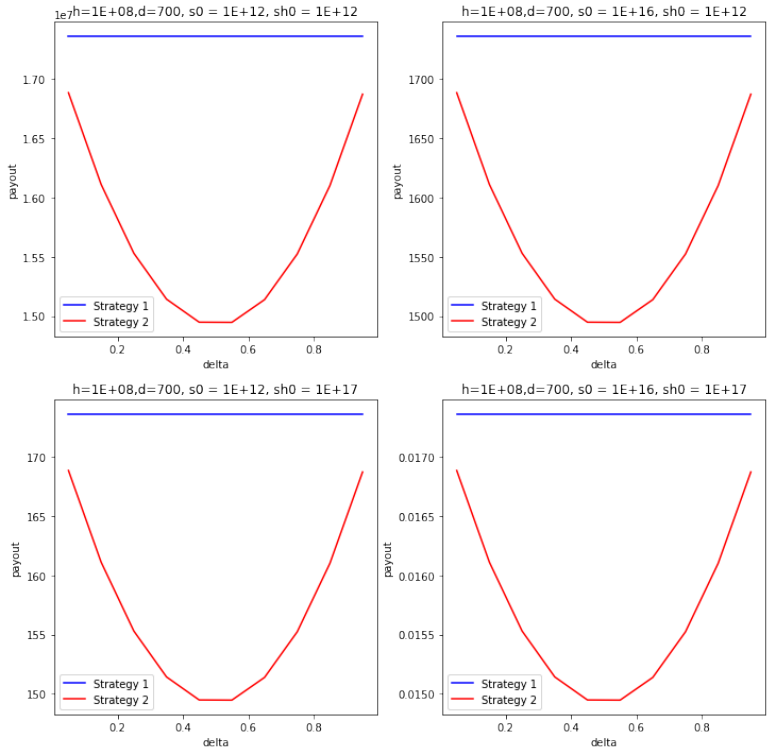

Figure 4: Moving δ for different values of $s_0, \texttt{sh}[0]$

recall that this implies that no stakes are started or ended during the run of Strategies 1 or 2. Note that this implies that `shareRate` is not updated for Strategy 1 (it always has the value $\texttt{sh}[0]$) and it is only updated for the second stake of Strategy 2 (after the first stake ends). This implies that the value `shareRate` is unusually low during the accrual of the stakes.

In Figure (5), we set $s_0 = 1e7$, $\texttt{allocSupply}[0] = 1.8e19$ (which holds the value of $\texttt{TOTAL\_SUPPLY} + \text{g.}\texttt{\_lockedHart}$ on day 0) and we draw two lines:

- The blue line describes stake-payouts according to the long strategy (Strategy 1) whereby h hearts are staken for $d = 700$ days. The x-axis describes the input stakes and the y-axis describes the payout.
- The red line describes stake-payouts according to a short strategy (Strategy 2) whereby h hearts are staken for $d = 350$ days, and once the stake return of$h + F(h, 350)$ is minted to the user, this is immediately re-staken for 349 days, and then the stake is ended. The x-axis describes the input stakes h and the y-axis

describes the (total) payout.

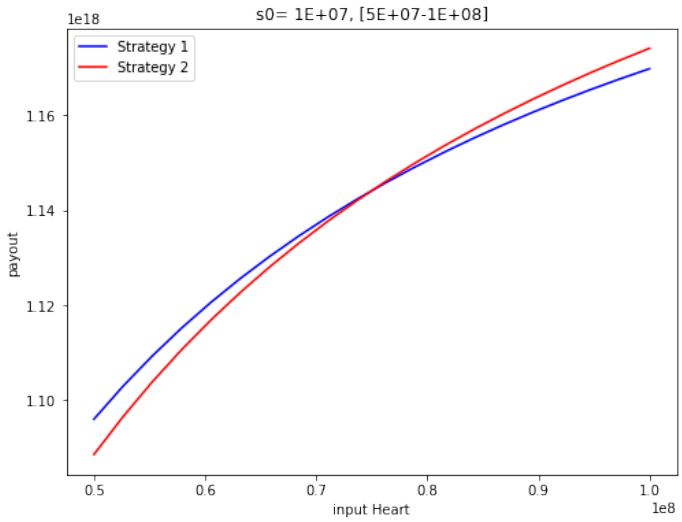

Figure 5: Simple example

Here x moves in $[0.5e8, 1e8]$ so that the crossing of curves can be visualized more clearly. Note that Strategy 2 is better than Strategy 1 for some values of h.

We understand these are parameter values not one that are likely to happen, e.g., the payout is over $1e18$ for an initial stake of $1e8$! Note, for example, that after the stake in Strategy 1 the global variable g._shareRate is updated from $1e5$ to values in the range of $1e16$! This points to the fact that a value of sh[0] as low as $1e5$ is unlikely to be found and maintained for 700 days.

Discussion

One may try to understand when does this happen: under what conditions Strategy 2 (short re-compounding) may win over Strategy 1 (long)? When competing actors are executing strategies 1 and 2 (concurrently), then longer pays better. So, for Strategy 2 ("short") to win, it is required that the user runs alone–among other requirements. That is, even acting alone is not enough, and it is required that sh[0] holds 'low' values. The graph below provides some insight into what are these 'low' values: Under these conditions, when sh[0] is somewhere between $1.2e8$ and $1.5e8$ the anomalous situation is gone and Strategy 1 wins over Strategy 2.

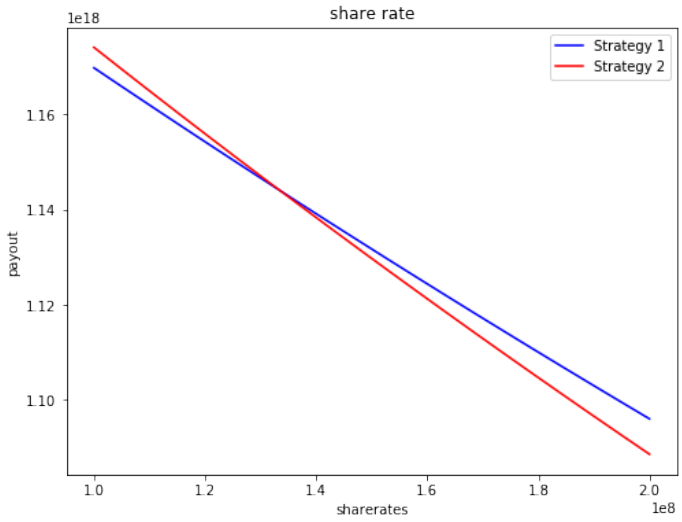

Figure 6: `shareRate` analysis

On the other hand, if anytime before these strategies run, another user would end a somehow similar stake (e.g., $h = 1e12, d = 700$), then `shareRate` is updated to a value over $1e16$. Hence, this sets us in the situation discussed in the previous section where longer always pays better than shorter.

Appendix

We have claimed that, in order to prove that long stakes are better than any combination of short-recompounding stakes, it suffices to show this for a combination of two shorter stakes (always, when using the same amount of resources).

We show this by showing that if there exist h and $(d_1, \ldots, d_n), d$ such that

$$F(h; d) < F(h; d_1, \ldots, d_n)$$

with $d \geq d_1 + \cdots + d_n + (n-1)$ for some $n \geq 2$, then there exist d'_1, d'_2 such that $d \geq d_1 + d_2 + 1$ and $F(h; d) < F(h; d'_1, d'_2)$.

Sketch of proof.- Fix h and assum that there exists at least one set of integers $d, d_1, \ldots, d_n$ with $d \geq d_1 + \cdots + d_n + (n-1)$ and $F(h; d) < F(h; d_1, \ldots, d_n)$. If there exists more than one such strategy, there must be one with n being minimum, i.e., if we let $(h, d); (h; d_1, \ldots, d_n)$ denote these strategies, then any other strategy $(h, e_1, \ldots, e_m)$ with $F(d, h) < F(h, e_1, \ldots, e_m)$ and $d \geq e_1 + \cdots + e_m + (m-1)$ it must be that $m \geq n$.

If $n = 2$ we have nothing to prove. Assume $n > 2$. Since n was minimal with this property, it follows that if we compare $F(h, d_1 + d_2 + 1)$ with $F(h; d_1, d_2)$ then it must be that

$$F(h; d_1, d_2) \leq F(h, d_1 + d_2 + 1) \tag{12}$$

(Else, this would contradict the minimality hypothesis.)

On the other hand, note that

$$F(h; d_1, d_2) = F(h; d_1) + F(F(h; d_1), d_2)$$

so that

$$\begin{aligned} F(h; d_1, d_2, \ldots, d_n) &= F(h; d_1) + F(F(h; d_1), d_2) + F(F(F(h; d_1), d_2); d_3 \ldots, d_n) \\ &= F(h; d_1, d_2) + F(F(h; d_1, d_2); d_3 \ldots, d_n) \\ &\leq F(h; d_1 + d_2 + 1) + F(F(h; d_1, d_2); d_3 \ldots, d_n) \\ &\leq F(h; d_1 + d_2 + 1) + F(F(h; d_1 + d_2 + 1); d_3 \ldots, d_n) \\ &= F(h; d_1 + d_2 + 1, d_3 \ldots, d_n) \end{aligned} \tag{13}$$

where the fact that

$$F(F(h; d_1, d_2); d_3, \ldots, d_n) \leq F(F(h; d_1 + d_2 - 1); d_3, \ldots, d_n)$$

follows from (12) and the fact that $F(H, D)$ is increasing in the variable H for any D (i.e., if $h_1 < h_2$ then $F(h_1, D) < F(h_2, D)$ for any D).

This, rather obvious fact, that a bigger stake implies bigger payouts can be proved by computing the (formal) derivative of $F(H, D)$ with respect to the first variable and ensuring it is positive for arbitrary values of H and D. The proof is technical, un-illustrative and left outside of this report.