.

Dynamic TAO

BIT001

Yuma Rao
Dr. Nick
0xcacti

In this document, we present Dynamic TAO (DTAO): a market-driven mechanism designed to replace Bittensor's current emission allocation model. In particular, Dynamic TAO derives emission values from the market prices of subnet-specific tokens that trade against TAO on Constant Product AMMs. This fundamentally transforms the existing validator-weighting system. More specifically, by introducing AMM-based emission allocation, we extend the subnet valuation system beyond Bittensor's validator set to the rest of the Bittensor ecosystem, including miners and speculators. By replacing the current subnet valuation model with an open market, we make the subnet valuation process more efficient, increase coordination costs for malicious participants aiming to manipulate emissions, and eliminate the apathetic oligarchic voting system.

This document will be presented in five sections:

- Section 1: Motivation
- Section 2: What is DTAO
- Section 3: Technical Overview
- Section 4: A Mathematical Model
- Section 5: Appendix

Section 1: Motivation

Bittensor incentivizes the production and distribution of digital commodities by emitting newly minted TAO in each block to a set of subnets. This TAO is distributed to the subnets according to what is collectively referred to as the emission vector $E \to [E_1, E_2, ..., E_n]$. Within each subnet, this emission is divided across three participant groups in the corresponding proportion:

- Validators (41%)
- Miners (41%)
- Subnet Owners (18%)

The details of the calculation are not important here, but roughly speaking, the emission vector is determined by the following calculation (where i stands for the i^{th} subnet):

$$E_i = \sum_j \binom{j^{th} \text{ validator's}}{\text{staked TAO}} \binom{j^{th} \text{ validator's favorability}}{\text{weight of } i^{th} \text{ subnet}}$$

The driving term in the above equation is the validator's favorability towards each subnet. Thus, these favorabilities essentially constitute a form of voting. Perhaps ideally, the weight vector resulting from this voting process would reward the subnets that are the most deserving, or those with the highest capital requirements. Problematically, the current system relies entirely on validators to manually assess and determine subnet value through their voting power. This creates a fundamental scaling problem: as the number of subnets grows, validators become increasingly unable to thoroughly evaluate each subnet's contribution to the network. The sheer volume of subnets requiring assessment often leads to validator apathy, where thorough evaluation becomes practically impossible.

This apathy can be further exacerbated by the lack of any meaningful consequences for validators who fail to maintain active and accurate weighting patterns. Indeed, the system provides no direct incentive for validators to regularly update their weight assignments or carefully consider their voting decisions. Even worse, the system incentivizes behavior that need not align with any ideal hypothetical weight distribution across subnets. As an example of a trivial way of manipulating the mechanism, validators may put more weight on subnets they actively validate upon in order boost their rewards (even if these are suboptimal subnets). Indeed, it would be merely altruistic for validators to provide weight to subnets on which they do not actively validate.

Another form of manipulation involves subnets owners offering revenue sharing agreements or other inducements to validators in exchange for larger weights on their subnet. Insidiously, this is a win-win scenario for the validators and subnet owners.

Fundamentally, the current incentive mechanism, with all of its flaws, can lead to less deserving subnets being rewarded in place of the more deserving subnets that would in fact accrue greater value to the Bittensor ecosystem. This is the motivation for introducing DTAO.

Section 2: What is DTAO

At its core, Dynamic TAO (DTAO) is an upgrade to the Bittensor chain that replaces the previous emission logic with an intelligent, market-driven mechanism that can be used to determine token emissions. Operationally, we introduce subnet tokens (which we will informally refer to as Alpha1, Alpha2, Alpha3, etc.) through which miners, validators, and subnet owners will earn rewards. In so far as Bittensor enables the decentralized production and distribution of digital commodities, subnet tokens serve as the medium through which the network intelligently *values* these commodities. To facilitate this valuation, we also introduce subnet pools, which are Constant Product AMMs (described in details in section 3.1) through which users can stake TAO to receive subnet tokens and vice versa. The price discovery that occurs in these subnet pools functions as a view into how the Bittensor network values each subnet. Critically, this is exactly what we want, i.e. a scalable mechanism for valuing subnets that does not rely on a priviledged, corruptible set of validators. Specifically, we will use the subnet token prices to guide the emission process.

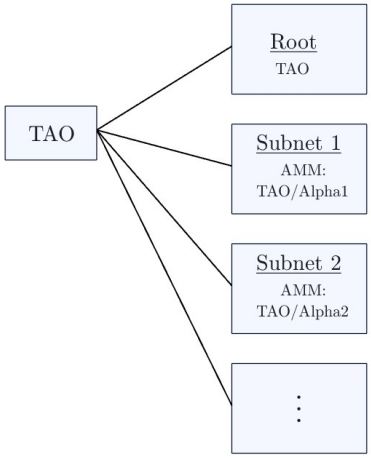


Figure 1: The Bittensor Subnets.

DTAO represents a fundamental shift towards true market democracy in the valuation of subnets. Where the previous system concentrated power in the hands of a few validators, DTAO opens participation to anyone willing to stake TAO in subnet pools. This broader participation not only makes the system more resistant to manipulation (try bribing a vast number of market participants for an extended period of time) but also ensures that subnet valuations reflect the collective wisdom of the market rather than the potentially biased views of a small group. Moreover, this market-driven approach scales naturally with network growth - as more subnets are added, the market simply expands to accommodate them, without degrading the quality of price discovery. Of course, this upgrade leads to a number of downstream changes, the core of which are discussed in detail in section (3.1) - (3.5), and the rest discussed in the appendix.

.

Section 3: Technical Overview

Section 3.0: A Look at the Components

In this section, we will look at some of the technical aspects of the DTAO system in a little more detail. There are many components to go over, so we will divide this section into five subsections, according to the following:

(3.1) the AMM governing the subnet pools
(3.2) the way in which new liquidity is injected
(3.3) how TAO may be staked/unstaked from a subnet
(3.4) the way that emissions are issued to users
(3.5) the halving schedule

Additionally, there are several (less immediately relevant) technical sections reserved for the appendix.

Section 3.1: Subnet AMMs

Each subnet has its own liquidity pool allowing for swaps (and hence for price discovery) between the subnet-specific tokens and TAO. These pools will conduct swaps using the standard *constant product* AMM, and for this reason, we include a quick reminder of the relevant mechanics.

In a constant product AMM, we consider a liquidity pool containing two types of tokens, and we denote the total reserves of each by x and y. Swaps are conducted in a way so as to maintain the product of the token reserves. So, for a swap that changes the reserves by amounts $(\Delta x, \Delta y)$, we must satisfy $(x+\Delta x)(y+\Delta y) = xy$. This results in the set of viable states existing on a hyperbola given by the equation $xy{=}L^2$, for some liquidity constant L. Moreover, it is easily shown that the spot price p (i.e. the exchange rate $\Delta y/\Delta x$, for small values) is given by the ratio of the reserve amounts $p{=}y/x$. These relationships, in turn, allow us to write the reserves as $x = L/\sqrt{p}$ and $y = L\sqrt{p}$.

New liquidity can be added to a constant product pool in any number of ways. However, if we wish to maintain the current spot price, then the added amounts $(\Delta x, \Delta y)$ must, themselves, be at the ratio of the current spot price. In other words, if we have $y/x = p$ and $\Delta y/\Delta x = p$, then we will also have $(y+\Delta y)/(x+\Delta x) = p$.

For the remainder of this document, the subscript 'i' will be reserved for the i^{th} subnet, and we will denote the subnet pool reserves by the following:

$$\alpha_i = \text{amount of subnet token in the pool} \tag{1}$$

$$\tau_i = \text{amount of TAO token in the pool} \tag{2}$$

Indeed, we will sometimes informally refer to the subnet token simply as 'Alpha'. Moreover, we let p_i denote the subnet token price denominated in TAO (i.e. the conversion from Alpha to TAO value). Then, from our discussion of the constant product AMM, we may write

$$p_i = \tau_i/\alpha_i, \tag{3}$$

i.e the subnet token price p_i (price of Alpha in terms of TAO) is the ratio of subnet pool reserve quantities.

Section 3.2: Injections

At each block, an amount of Alpha and TAO tokens are to be added to the pool reserves. We call this an *injection*, and we denote these injection quantities by $(\Delta\alpha_i, \Delta\tau_i)$. These injection amounts are meant to reflect the relative performance of the subnets. There is no *correct* way to do this, but it is natural to imagine the subnet tokens as being speculative instruments, such that their prices correlate with subnet performances. We make the choice to *define* the subnet emissions along this line of reasoning.

To be precise, we fix a certain amount of total TAO to be emitted at each block, denoted $\Delta\overline{\tau}$, and we begin with the idea that this quantity $\Delta\overline{\tau}$ should be divided up among the subnets in proportion to their Alpha prices. In other words, the TAO injection for the i^{th} subnet $(\Delta\tau_i)$ would be given by the following:

$$\Delta\tau_i = \frac{p_i}{\sum_j p_j} \times \Delta\overline{\tau} \tag{4}$$

When doing the injection, we don't want to artificially alter the subnet pool price. In other words, we should choose the corresponding Alpha injection $(\Delta\alpha_i)$ so that it maintains the ratio given in expression (3). Specifically, this means that we must have the relation $\Delta\alpha_i = \Delta\tau_i/p_i$, which then results in the following:

$$\Delta\alpha_i = \frac{1}{\sum_j p_j} \times \Delta\overline{\tau} \tag{5}$$

However, we will want to modify this formula based on the following observation; *if the sum of prices $\sum p_j$ becomes small, the alpha injection $\Delta\alpha_i$ can grow without bound.* In order to prevent runaway inflation of the subnet tokens, we modify formula (5) by putting a *cap* on the amount of Alpha emitted, which we will denote by $\Delta\overline{\alpha}_i$ (note that this cap depends on the particular subnet i, depending on where the subnet is in its halving schedule, see section 3.5). Our injection formulas can then be summarized as follows:

$$\Delta\tau_i = \frac{p_i}{\sum_j p_j}\Delta\overline{\tau} \tag{6}$$

$$\Delta\alpha_i = \min\left\{\frac{\Delta\overline{\tau}}{\sum_j p_j},\, \Delta\overline{\alpha}_i\right\} \tag{7}$$

We note two immediate consequences from these formulas:

$$\textstyle\sum_i \Delta\tau_i = \Delta\overline{\tau} \tag{8}$$

$$\Delta\alpha_i \leq \Delta\overline{\alpha}_i \tag{9}$$

Thus, we have firm upper bounds on the emission amounts.

The value of $\Delta\overline{\tau}$ is initialized to be 1, to maintain the current emission schedule of 1 TAO per block. Similarly for subnet tokens, we will take $\Delta\overline{\alpha}_i = 1$ when each subnet is initialized, though all of these values are subject to halve according to the halving schedule (again, see section 3.5).

It should be noted that the '*min*' statement in (7) has the following effect: when the sum of all prices drops below a subnet-specific threshold (i.e if $\sum_j p_j < \Delta\overline{\tau}/\Delta\overline{\alpha}_i$), then less Alpha is emitted than would be needed to maintain pool price, and this drives the price of the pool *up*. This will be critical feature in the early days of low liquidity when pool prices are more sensitive to movement.

Our injection formulas from (6)-(7) are presented in a way that is meant to be both readable and mathematically motivated. However, the way in which these quantities are computed in code is actually slightly different (while of course producing the same values). For the interested reader, we present the pseudo-code below. The variables `tao_in` and `alpha_in` represent the amount of tokens being injected into the subnet pool reserves, while the variable `alpha_out` represents the alpha to be emitted to the subnets, specifically as miner incentives and validator dividends. (One may note, the 'inject' function not only injects liquidity into subnet pools, but it also delivers the `alpha_out` to the relevant parties.)

```
sum_of_prices = sum(subnet_prices)
for subnet in subnets:
    tao_in = subnet.price/sum_of_prices
    alpha_in = tao_in/subnet.price
    if alpha_in > max_alpha_emission:
        alpha_in = max_alpha_emission
    alpha_out = max_alpha_emission
    subnet.inject(
        tao_in,
        alpha_in,
        alpha_out
    )
```

Figure 1: Injection pseudo-code

Lastly, it should be noted that when using the injection formulas in practice, we will actually use the *exponentially weighted moving average* (EMA) in place of the actual subnet pool price p_i. This is done for several reasons:

- to discourage potential malicious attacks that can operate through price manipulation
- to smoothen out otherwise volatile price movements that may occur for low liquidity pools.

Section 3.3: Staking/Unstaking

If a user swaps some TAO on a subnet pool to obtain some subnet token (Alpha), then we say that the user *stakes* the TAO to the subnet. Similarly, if a user redeems their subnet token by swapping it for TAO, we say that the user *unstakes* TAO. Thus, the mechanics of staking and unstaking are relatively straightforward, as the rules for a constant product AMM are well established.

It should be noted, however, that different subnet tokens cannot be exchanged directly with each other. For example, if a user wishes to exchange some subnet token $\Delta\alpha_i$ for other subnet token $\Delta\alpha_j$, they must first *unstake* some TAO from subnet i (thereby relinquishing their $\Delta\alpha_i$) and then they must *stake* that TAO on subnet j to obtain the desired amount $\Delta\alpha_j$.

Lastly, it is worth noting that, unlike in typical AMMs, there will be no fees taken on the swaps, as there will be no liquidity providers to claim such fees. In other words, all liquidity of Alpha and TAO tokens is provided by the emission process.

.

Section 3.4: Subnet Emissions

Whereas validators, miners, and subnet owners were previously rewarded for their participation in the network through TAO, they will now be rewarded in Alpha. To accomplish this, we will have an additional quantity of Alpha, denoted $\Delta\alpha_i'$ that is emitted at each block, and this quantity is to be divided up and distributed to users. For simplicity, we take $\Delta\alpha_i'$ to be equal to the value $\Delta\overline{\alpha}_i$ (i.e. the maximum amount of Alpha injected into the pool per block, as previously defined):

$$\Delta\alpha_i' = \Delta\overline{\alpha}_i \tag{10}$$

Then the precise way in which we divide this quantity up can be described as follows:

- The Alpha emitted $\Delta\alpha_i'$ is divided up into three pieces, according to the ratio 41:41:18.
  - 18% go to subnet owners
  - 41% go to miners
  - 41% go to validators
- Next, we consider the the following quantities:
  - α_i^o = the *alpha outstanding* is the total supply of Alpha held by users (not pool reserves)
  - τ_0 = the total TAO staked in the root subnet
  - γ = a freely chosen parameter that we call the *tao weight*, to be discussed momentarily

  With these quantities, we define something called the *root proportion*, denoted by r, and defined as

$$r_i = \frac{\gamma\,\tau_0}{\gamma\,\tau_0 + \alpha_i^o} \tag{11}$$

- We use this ratio r_i to split the validator dividends into two separate portions:
  - the fraction r_i is distributed to root stakers
  - the fraction (1-r_i) is given to subnet validators

This process can be summarized in the following figure:

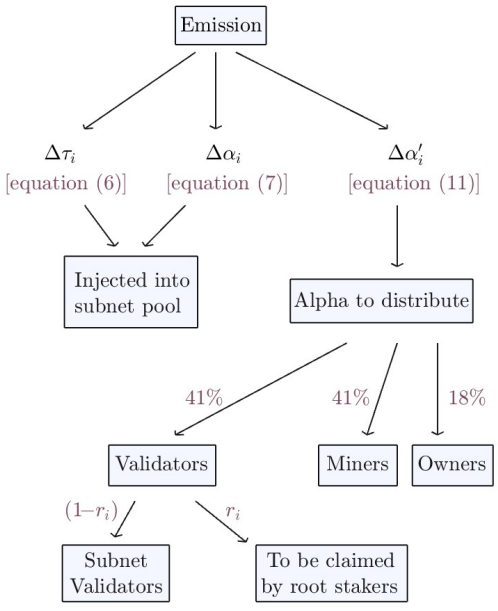


Figure 2: Flow chart of emission quantities

.

Now, one may wonder how we actually choose the value of this *tao weight* parameter. Generally speaking, the tao weight is meant to help smoothen the transition to the current DTAO design. In particular, there are a few guiding principles and motivations behind this parameter:

- We want to eventually transition from the current economic consensus to one fully governed by Alpha, but there are dangers in transitioning too quickly. In particular, the shifting supply and relatively low liquidity of the subnet tokens will drastically change in the first few months, and this can severely upset the existing consensus. However, with the tao weight parameter, we can slow this transition by diverting emissions to the root network initially, and we can keep the economic weight from shifting too fast. In the following figure, we plot the percent ownership of Alpha held by the root validators over time. This quantity naturally decreases over time, but the steepness of the curve varies depending on the specific chosen value of the tao weight:

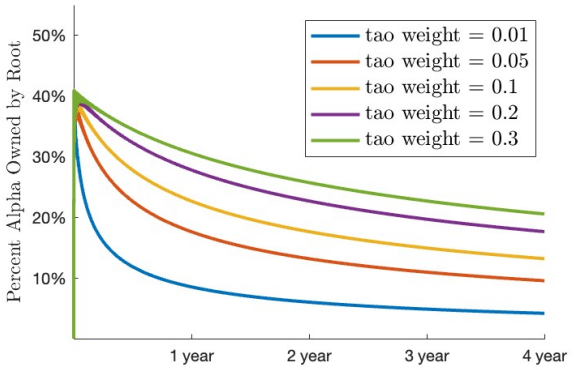


Figure 3: Root percent ownership of Alpha over time

- Though we do not discuss the consensus mechanism here, we note that the root proportion r is also used to blend the stake weight (used in Yuma consensus) between TAO and Alpha. By tuning the value of the tao weight, we effectively have some control over the period where consensus transitions from being TAO dominated to being Alpha dominated. For example, if we instantly transition to a state of relying entirely on the Alpha stake for consensus, then as pools launch with low initial liquidity, it would be trivial for validators to try to quickly acquire Alpha and manipulate consensus on a given subnet. Thus, the tao weight allows us to prevent situations like this.

- The tao weight can prevent early Alpha holders from receiving absurdly high APYs. For example, note that *if* the tao weight is equal to zero ($\gamma=0$), then expression (12) tells us that the root proportion will also be zero ($r=0$). Thus, from the flow chart above, it is clear that root stakers who do not own Alpha will receive *no dividends* at all in this case. One can easily work out that this results in a situation where the initial Alpha holders receive runaway dividends with shockingly high APYs (see figure 4).

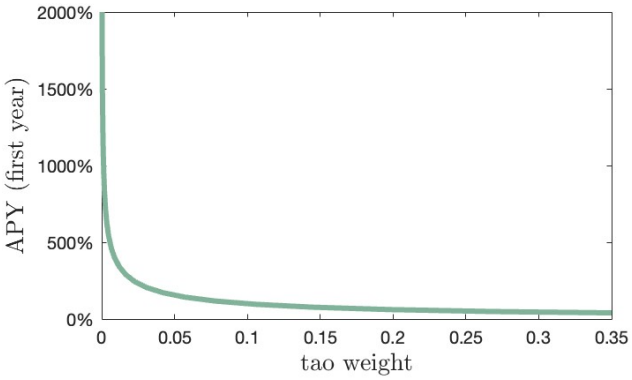


Figure 4: APY for early Alpha holders v.s. tao weight

Section 3.5: Halving Schedule

The TAO halving schedule will remain unchanged from the previously established halving schedule. Specifically, the amount of TAO emitted per block is always equal to the quantity $\Delta\overline{\tau}$, and this quantity is subject to halve every time the accumulated TAO supply reaches certain supply thresholds. This cause the emission rate to halve roughly every four years (the details are provided in section 5.2).

New Alpha tokens will also follow the same halving schedule, with the emission amount $\Delta\overline{\alpha}_i$ being halved when reaching the same supply thresholds as TAO does (see appendix). Of course, the growth rate of Alpha supply can be up to *double* the value of $\Delta\overline{\alpha}_i$, due to the fact that we inject some Alpha for the pool reserves (which is less than or equal to $\Delta\overline{\alpha}_i$) as well as the emission amount (equal to $\Delta\overline{\alpha}_i$) for the rewards. Thus, while the growth rate of Alpha supply will be faster in time than that of the TAO supply, all halving events never the less occur at the same set of supply thresholds, and therefore every token supply approaches the same asymptotic value: 21,000,000. The following figure demonstrates this:

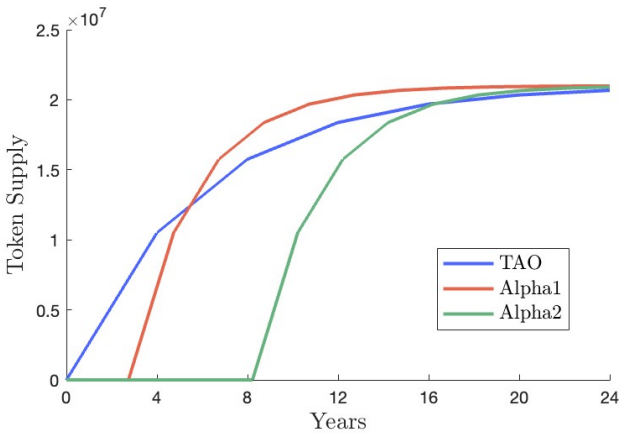


Figure 5: TAO and Alpha supply growth over time

Lastly, we note that as a consequence of the tapering rate of token supply growth, subnets that launch sooner in time will benefit from periods of faster liquidity growth in their subnet pools. Conversely, subnets that launch later in time must be content with slower liquidity growth.

Section 4: Mathematical Toy Model

In this section, we construct a simple mathematical toy model of the DTAO system, from which we may extract useful formulas for long term predictions. We will begin by establishing the governing rules for the model. We then make a few simplifying assumptions/approximations, and arrive at some useful formulas. Finally, we compare these formulas to some numerical simulations.

Section 4.1: Establishing a Mathematical Model

To set the stage, we will assume that we have N-many subnets (indexed by 'i') with constant product AMMs to swap between alpha and tao tokens. The token reserve quantities are denoted by $\{\alpha_i, \tau_i\}$, as functions of time:

$$\tau_i(t) = \text{TAO reserves at time } t \text{ } (i^{th} \text{ subnet}) \tag{12}$$

$$\alpha_i(t) = \text{Alpha reserves at time } t \text{ } (i^{th} \text{ subnet}) \tag{13}$$

As constant product pools, they obey the following:

$$p_i(t) = \tau_i(t)/\alpha_i(t), \tag{14}$$

$$L_i(t) = \sqrt{\alpha_i(t)\tau_i(t)}, \tag{15}$$

where p_i is the price of *Alpha* in term of TAO, and L_i is the liquidity scale factor for the pool. At regular intervals (blocks), the reserves update according to section 3.2:

$$\Delta\tau_i(t) = \frac{p_i(t)}{\sum_j p_j(t)}\Delta\overline{\tau} \tag{16}$$

$$\Delta\alpha_i(t) = \min\left\{\frac{\Delta\overline{\tau}}{\sum_j p_j(t)}, \Delta\overline{\alpha}_i\right\} \tag{17}$$

The pool reserves are altered anytime a swap is executed, and these swaps then determine the pool price. However, for our mathematical model, we will suppose that the price trajectories $p_i(t)$ are *given* (at least, probabilistically), and thus the pool reserves are determined implicitly. In particular, we note that while the liquidity factor L_i is given by (16), it will also be determined simply by the cumulative amount of tokens injected. This is because swaps do not alter the liquidity constant (indeed, that is the *whole point* of the constant product), and so only the injection can determine L. Thus, we define the cumulative token injections (α_i^*, τ_i^*) by the following:

$$\tau_i^*(t) := \textstyle\sum_s^t \Delta\tau_i(s) \quad \text{(cumulative injected tao)} \tag{18}$$

$$\alpha_i^*(t) := \textstyle\sum_s^t \Delta\alpha_i(s) \quad \text{(cumulative injected alpha)} \tag{19}$$

With these defined, we can then restate L:

$$L_i(t) = \sqrt{\left[\alpha_i(0) + \alpha_i^*(t)\right]\left[\tau_i(0) + \tau_i^*(t)\right]} \tag{20}$$

Finally, we define one more bit of notation for convenience. Specifically, we denote the sum of prices by $S(t)$:

$$S(t) := \textstyle\sum_i p_i(t) \tag{21}$$

Then our deltas in (17)-(18) can be rewritten as

$$\Delta\tau_i(t) = p_i(t)\frac{\Delta\overline{\tau}}{S(t)} \tag{22}$$

$$\Delta\alpha_i(t) = \min\left\{\frac{\Delta\overline{\tau}}{S(t)}, \Delta\overline{\alpha}_i\right\} \tag{23}$$

which will be slightly more convenient later on.

Section 4.2: Three assumptions/approximations

For our first simplifying assumption, *we will assume that the prices $\{p_i(t)\}$ each evolve according to Geometric Brownian Motion (GBM)*. Specifically, this means that for each subnet, we specify

- an *initial price* $p_i(0)$
- a *drift* parameter μ_i
- a *volatility* parameter σ_i

and then the price $p_i(t)$ at time t will be distributed with the following log-normal probability density:

$$p_i(t) \sim \frac{1}{\sqrt{2\pi\sigma_i^2 t}}\frac{1}{p}\exp\left(\frac{-\left(\log[p/p_i(0)] - \mu_i t\right)^2}{2\sigma_i^2 t}\right) \tag{24}$$
$$(\text{for } 0 < p < \infty)$$

Now, for a given set of parameters $\{p_i(0), \mu_i, \sigma_i\}_i^N$, our first goal will be to compute the *expected values* of (α_i^*, τ_i^*), i.e the cumulative injected tokens, up to some time T:

$$E[\tau_i^*(T)] = \text{expected cumulative TAO injected} \tag{25}$$

$$E[\alpha_i^*(T)] = \text{expected cumulative Alpha injected} \tag{26}$$

To compute these, we accept the following approximation: *we suppose the sums in (19)-(20) can be approximated as integrals*. For example, with the expected cumulative Alpha in (27), we develop as follows:

$$\begin{aligned} E\left[\alpha_i^*(T)\right] &= E\left[\textstyle\sum_t^T \Delta\alpha_i(t)\right] \\ &= \textstyle\sum_t^T E\left[\Delta\alpha_i(t)\right] \\ &= \textstyle\sum_t^T E\Big[\min\{\Delta\overline{\tau}/S(t), \Delta\overline{\alpha}_i\}\Big] \\ &= \textstyle\sum_t^T E\left[\min\left\{\frac{(\Delta\overline{\tau}/\Delta t)}{S(t)}, (\Delta\overline{\alpha}_i/\Delta t)\right\}\right]\Delta t \\ &\approx \int_0^T E\Big[\min\{\Delta\overline{\tau}/S(t), \Delta\overline{\alpha}_i\}\Big]dt \end{aligned} \tag{27}$$

This approximation should be reasonable since the sums happen over frequently occurring blocks, and are hence nearly continuous, while the rates $(\Delta\overline{\tau}/\Delta t)$ and $(\Delta\overline{\alpha}_i/\Delta t)$ are constant and can simply be replaced with $\Delta\overline{\tau}_i$ and $\Delta\overline{\alpha}_i$. Next, we approximate the expected cumulative TAO:

$$E\left[\tau_i^*(T)\right] \approx \int_0^T E\Big[p_i(t)/S(t)\ \Delta\overline{\tau}\Big]dt \tag{28}$$

Unfortunately, the appearance $S(t)$ in the integrand of (28) and (29) makes them exceedingly difficult to calculate. In particular, $S(t)$ depends on the entire set of prices $\{p_i(t)\}$, and so the expectation $E[\cdot]$ must be taken over the *joint* probability distribution of all of the N-many price paths. Instead, we make our next approximation; *we will replace $S(t)$ with its expectation $E[S(t)]$*, denoted by $\overline{S}(t)$:

$$\overline{S}(t) := E[S(t)] = E\Big[\textstyle\sum_i p_i(t)\Big] = \textstyle\sum_i E\big[p_i(t)\big] \tag{29}$$

The substitution $S(t) \approx \overline{S}(t)$ is not completely unjustified; $S(t)$ is a sum of approximately normal random variables, and by the generalized Central Limit Theorem, it will also be approximately normally distributed around its mean, and with a relatively small variance over reasonably small time scales. Indeed, we will ultimately check it numerically.

Now, to compute the expectation of each price $E\big[p_i(t)\big]$, we use the probability density given in (25).

$$E\big[p_i(t)\big] = \int_0^\infty \frac{p}{\sqrt{2\pi\sigma_i^2 t}} \frac{1}{p} \exp\left(\frac{-\big(\log[p/p_i(0)] - \mu_i t\big)^2}{2\sigma_i^2 t} \right) dp \quad (30)$$

With the change of variables $x := \log[p/p_i(0)]$, the integral in (31) can then be written as

$$E\big[p_i(t)\big] = p_i(0) \int_{-\infty}^\infty e^x \frac{1}{\sqrt{2\pi\sigma_i^2 t}} \exp\left(\frac{-(x-\mu_i t)^2}{2\sigma_i^2 t} \right) dx \quad (31)$$

We can evaluate (32) by use of the standard formula

$$\int_{-\infty}^\infty e^{cx} \frac{1}{\sqrt{2\pi a}} \exp\left(\frac{-(x-d)^2}{2a} \right) dx = \exp\left(cd + \frac{1}{2}ac^2 \right) \quad (32)$$

which can easily be derived by completing the square inside the exponential. Applying (33) to (32), we then find

$$E\big[p_i(t)\big] = p_i(0) e^{(\mu_i + \sigma_i^2/2)t} \quad (33)$$

Thus, expression (30) for $\overline{S}(t)$ then becomes

$$\overline{S}(t) = \textstyle\sum_i p_i(0) e^{(\mu_i + \sigma_i^2/2)t} \quad (34)$$

We note that, unlike the quantity $S(t)$, the quantity $\overline{S}(t)$ is *not* a random variable distributed with GBM, but is instead a deterministic function of t. Thus, using our third approximation, we replace $S(t)$ with $\overline{S}(t)$ in (28)-(29) and we find that it can now slip outside the expectation value:

$$\begin{aligned} E\left[\tau_i^*(T)\right] &\approx \int_0^T E\Big[p_i(t)/S(t)\ \Delta\overline{\tau}\Big] dt \\ &\approx \int_0^T E\big[\,p_i(t)\,\big] \Delta\overline{\tau}/\overline{S}(t)\, dt \quad (35) \\ E\left[\alpha_i^*(T)\right] &\approx \int_0^T E\Big[\ \min\{\Delta\overline{\tau}/\overline{S}(t), \Delta\overline{\alpha}_i\}\Big] dt \\ &\approx \int_0^T E\big[\,1\,\big]\, \min\{\Delta\overline{\tau}/\overline{S}(t), \Delta\overline{\alpha}_i\}\, dt \quad (36) \end{aligned}$$

We know that $E[1] = 1$, and we know $E[p_i(t)]$ from (34). Thus, (36)-(37) become

$$E\left[\tau_i^*(T)\right] \approx p_i(0) \int_0^T \Delta\overline{\tau}/\overline{S}(t)\ e^{(\mu_i + \sigma_i^2/2)t} dt \quad (37)$$

$$E\left[\alpha_i^*(T)\right] \approx \int_0^T \min\{\Delta\overline{\tau}/\overline{S}(t), \Delta\overline{\alpha}_i\}\, dt \quad (38)$$

Finally, we note that using

$$\min\{\Delta\overline{\tau}/\overline{S}(t), \Delta\overline{\alpha}_i\} = \frac{\Delta\overline{\tau}}{\max\big\{\overline{S}(t), \Delta\overline{\tau}/\Delta\overline{\alpha}_i\big\}} \quad (39)$$

we can rewrite (38)-(39) in a more explicit way:

$$E\left[\tau_i^*(T)\right] \approx \int_0^T \frac{\Delta\overline{\tau}\, p_i(0) e^{(\mu_i + \sigma_i^2/2)t}}{\Big\{\sum_j p_j(0) e^{(\mu_j + \sigma_j^2/2)t}\Big\}} dt \quad (40)$$

$$E\left[\alpha_i^*(T)\right] \approx \int_0^T \frac{\Delta\overline{\tau}}{\max\Big\{\sum_j p_j(0) e^{(\mu_j + \sigma_j^2/2)t},\ \Delta\overline{\tau}/\Delta\overline{\alpha}_i\Big\}} dt \quad (41)$$

Our final approximation will be made in an effort to compute the expected market cap for each subnet pool. When denominated in tao, the market cap of the i^{th} subnet pool, denoted $M_i(t)$ is given by

$$M_i(t) = \left[\begin{matrix}\text{total Alpha supply} \\ \text{at time } t\end{matrix}\right] \times \big[\text{price}\big] \quad (42)$$

The price at time t is of course given by $p_i(t)$. To calculate the total supply, we note that all the Alpha that can ever exist must come from the emissions. Thus, we simply need to add the cumulative emissions $\alpha_i^*(t)$ to any initial Alpha $\alpha_i(0)$, and this will give us our total supply. However, for good measure we will also include the emissions that go to users as rewards. These rewards were not included in our toy model so far, but it is a trivial thing to include. We know from section 3.5 that an amount $\Delta\overline{\alpha}_i$ is emitted every block, and so the total amount up to time t is $\Delta\overline{\alpha}_i t$ (assuming t is measured in blocks). Thus, our total expression for market cap will be given by

$$M_i(t) = \Big(\alpha_i(0) + \alpha_i^*(t) + \Delta\overline{\alpha}_i t\Big) p_i(t) \quad (43)$$

Our interest will be in the expected market cap at time $t = T$, and so we write the following:

$$\begin{aligned} E\big[M_i(T)\big] &= E\Big[\Big(\alpha_i(0) + \alpha_i^*(T) + \Delta\overline{\alpha}_i T\Big) p_i(T)\Big] \\ &= \big(\alpha_i(0) + \Delta\overline{\alpha}_i T\big) E\big[p_i(T)\big] + E\big[\alpha_i^*(T) p_i(T)\big] \quad (44) \end{aligned}$$

Now, the expression $E\big[\alpha_i^*(T) p_i(T)\big]$ would, in principle, be difficult to compute, as the expectation would be taken over the *joint* probability distribution of $\alpha_i^*(T)$ and $p_i(T)$. However, because of our approximation (30), the quantity $\alpha_i^*(T)$ is not actually a random variable. Thus we may distribute the expectation over $E[\alpha_i^*(T) p_i]$, obtaining

$$\begin{aligned} E\big[M_i(T)\big] &\approx \big(\alpha_i(0) + \Delta\overline{\alpha}_i T\big) E\big[p_i(T)\big] + E\big[\alpha_i^*(T)\big] E\big[p_i(T)\big] \\ &= E\big[p_i(T)\big] \Big(\alpha_i(0) + \Delta\overline{\alpha}_i T + E\big[\alpha_i^*(T)\big]\Big) \quad (45) \end{aligned}$$

Importantly, we see that expression (46) is written in terms of quantities that we have already worked out, namely (34) and (42). This will be our expression for the final market cap.

Summary So Far

Given prices $p_i(t)$ with GBM parameters $\{p_i(0), \mu_i, \sigma_i\}$, and injection per unit time b, the expected accumulated alpha and tao tokens $\{\overline{\alpha}_i(T), \overline{\tau}_i(T)\}$, as well as the subnet market caps $M_i(T)$, at some time T, are approximated by:

$$E\left[\tau_i^*(t)\right] \approx \int_0^T \frac{\Delta\overline{\tau}\, p_i(0) e^{(\mu_i + \sigma_i^2/2)t}}{\Big\{\sum_j p_j(0) e^{(\mu_j + \sigma_j^2/2)t}\Big\}} dt \quad (46)$$

$$E\left[\alpha_i^*(t)\right] \approx \int_0^T \frac{\Delta\overline{\tau}}{\max\Big\{\sum_j p_j(0) e^{(\mu_j + \sigma_j^2/2)t},\ \Delta\overline{\tau}/\Delta\overline{\alpha}_i\Big\}} dt \quad (47)$$

$$E\big[M_i(T)\big] \approx E\big[p_i(T)\big] \Big(\alpha_i(0) + \Delta\overline{\alpha}_i T + E\big[\alpha_i^*(T)\big]\Big) \quad (48)$$

Section 4.3: Numerical Verification

To test our formulas, we run some numerical simulations. In particular, the purpose of the simulations in this section is simply to test the validity of our approximate expectation values given in (47)-(49). One should note, these will not be thorough agent-based simulations of the DTAO system in its full generality, but rather they are simulations of our idealized mathematical model articulated by the update rules (17)-(18), driven by simulated Geometric Brownian Motion price movements.

In order to completely visualize the results, we start with an unrealistically low number of subnets at

$$N = 4 \text{ subnets}$$

We use select the following drift and volatility parameters to generate a variety of price trajectories:

$$\mu = [-4, 0, 10, 2] \times 10^{-8}$$
$$\sigma = [5, 6, 5, 6] \times 10^{-5},$$

over a period of T=4 years, and $\Delta\overline{\tau}$=$\Delta\overline{\alpha}_i$=1 for all i. We also note that this simulation does not include any halving events (indeed, the halving schedule was not factored in to our mathematical model from the previous section at all).

We now run 1,000 trials. For each trial, we generate four separate independent GBM price paths (one for each subnet) according to the values of μ and σ just given. First, we simply confirm the correct nature of our GBM by collecting the final prices for each subnet during each trial, and plot the distribution of final prices. For good measure, we also plot the log-normal distribution that we would mathematically expect for the probability density given in formula (25). The results are plotted in figure 6.

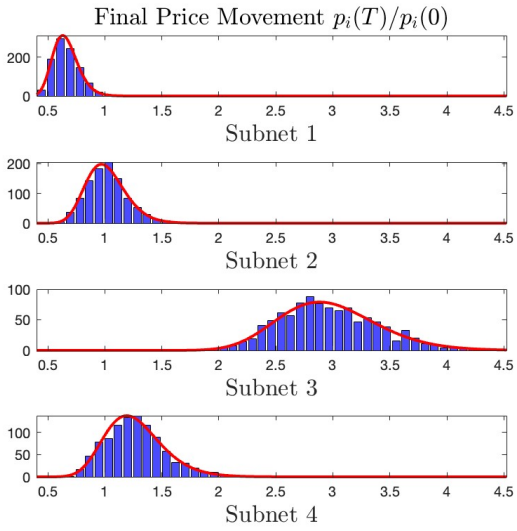


Figure 6: The distribution of final prices under GBM

We next plot the distributions for our three quantities of interest; the cumulative injected Alpha and TAO tokens, and the final market cap, shown in figure 7 below. Indeed, we have markers to indicate the mean values from our raw simulation data, along with the values that our formulas (47)-(49) would predict. We make three observations:

- The data means agree excellently with our formulas.
- The Alpha injections are identical for all subnets (this should be clear from the injection formula (7) - it does not depend on i, outside of $\Delta\overline{\alpha}_i$).
- The distributions of final prices, cumulative TAO, and market cap are all qualitatively similar.

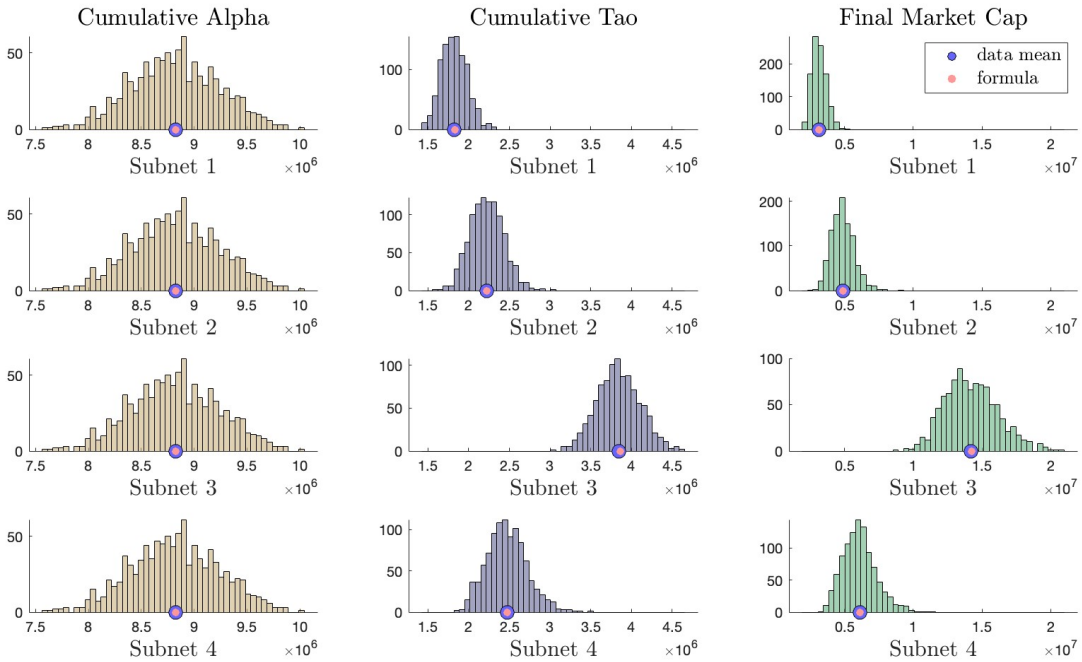
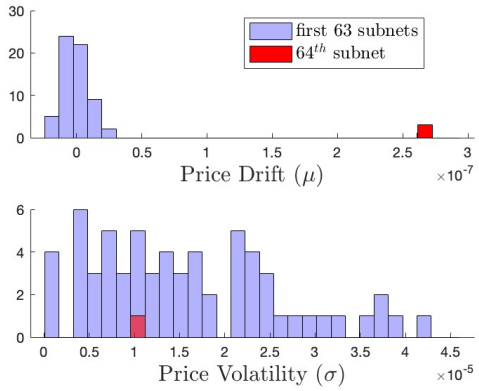


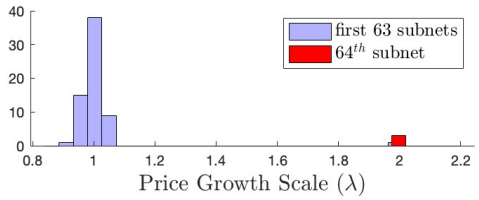
Figure 7: The distributions of final quantities

Section 4.4: Some Case Studies

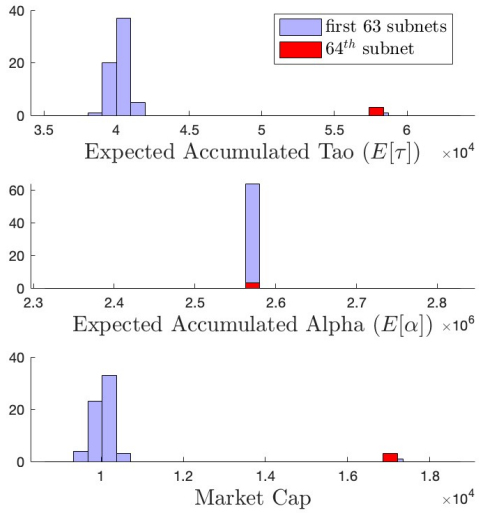
Now that we feel confident in our expectation formulas (47)-(49), let us use them in specific case studies. We begin with a simple scenario wherein one subnet has a noticeable upwards drift in price, while the other subnet prices remain relatively stable. Specifically, we will use $N=64$ subnets, over a period $T=1$ year, with block emission of $\Delta\overline{\tau}, \Delta\overline{\alpha}_i=1$. We assign a normal spread of negligible drift parameters to the first 63 subnets, and we give the 64^{th} a noticeably higher drift. Moreover, we assign similarly mild random volatility parameters to all subnets. We can visualize these parameter assignments below:



The drift parameter μ is hard to appreciate intuitively, but we convert it to the more relatable quantity $\lambda:=p(T)/p(0)$ (the price growth factor) by the equation $\lambda=e^{\mu T}$. Using this, we plot the equivalent spread of λ values:



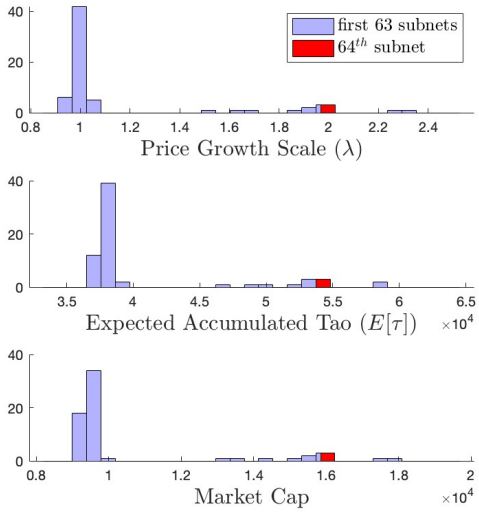
Thus, for example, our scenario corresponds with a subnet whose price *doubles* over the course of the year, while all other subnets move by about $\pm 5\%$. Moreover, now that we have established the GBM setting, we use our formulas to compute expected values after one year. We plot the cumulative Alpha and TAO, and the final market cap:



We can make a few observations from the previous figure. For our scenario with one subnet experiencing a doubling in price over the year, we can say the following:

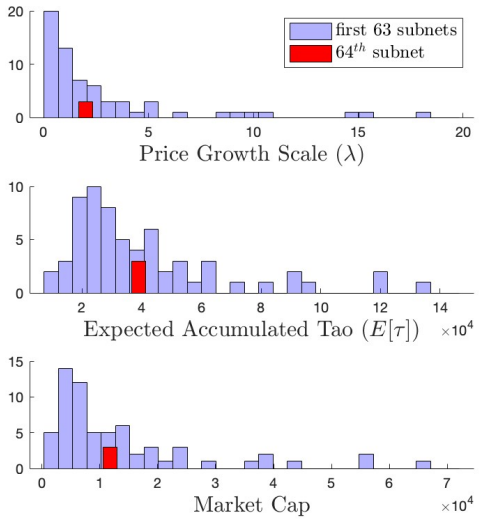
- It will receive about 45% more TAO tokens than the other subnets (note the $E[\tau]$ values of $\approx 40,000$ for the general subnets, compared to $\approx 58,000$ for our special subnet; this is a $\approx$45% increase)
- All subnets have the same expected cumulative alpha tokens - again, not a surprise, as we previously noted.
- Our subnet can expect about 70% greater market cap at the end of the year

What if we give our subnet some competition? Let's take our previous scenario but suppose that there is a group of, say, ten other subnets who have similarly desirable drift parameters. We find the following:



Not surprisingly, our subnet now earns slightly less TAO and overall market cap than in the previous example.

Finally, let us suppose that the subnets have a wide distribution of drift parameters, and that our subnet, while still experiencing a doubling of price, is situated somewhere in the middle of the pack. We find the following:



Armed with the formulas (47)-(49), the interested reader may investigate many more kinds of market trajectories and case studies beyond these simple scenarios.

Section 5: Appendix

Section 5.1: An APY Heuristic for Tao Weight

In this section we explore an interesting heuristic to help guide us towards a good choice for the tao weight γ. In particular, this heuristic will be based on comparing the APY earned by root stakers versus the subnet validators. As we saw in section 3.4, the amount of Alpha emission dedicated to the validators is given by $(0.41)\Delta\overline{\alpha}_i$. This is then split by the value of the root proportion r into the following two pieces:

$$\text{(root dividends from } i^{th} \text{ subnet)} = r(0.41)\Delta\overline{\alpha}_i \tag{49}$$

$$\text{(dividends for } i^{th} \text{ subnet validators)} = (1\text{-}r)(0.41)\Delta\overline{\alpha}_i \tag{50}$$

From our definition (12), we can write the following:

$$r_i = \frac{\gamma\,\tau_0}{\gamma\,\tau_0 + \alpha_i^o}, \quad (1\text{-}r_i) = \frac{\alpha_i^o}{\gamma\,\tau_0 + \alpha_i^o} \tag{51}$$

where we recall that

$$\gamma = \text{the tao weight} \tag{52}$$

$$\tau_0 = \text{TAO staked on root} \tag{53}$$

$$\alpha_i^o = \text{Alpha outstanding (not reserves)} \tag{54}$$

Now, for the Alpha outstanding term, we will make the following assumption; while Alpha may be coming in and out of the subnet pool, one could argue that if the subnet price remains relatively stable, then the Alpha outstanding should just be equal to the total cumulative Alpha emissions given out as dividends. Thus, starting at a value of $\alpha_i^o(0)$, over a period of t emissions, we employ the following simple model:

$$\alpha_i^o = \alpha_i^o(0) + t\Delta\overline{\alpha}_i \tag{55}$$

To get the total rewards over some specified period, we would need to sum over the full range of t appearing in (55). Moreover, to obtain the APY, we would divide the total rewards by the initial token held, i.e either τ_0 or $\alpha_i^o(0)$. We do this next, while substituting (51) and (55) back into the reward expressions in (49) and (50). Altogether, we get the following expressions for the APYs:

$$\text{APY}_i^{root} = \frac{1}{\tau_0}\sum_t \frac{\gamma\tau_0(0.41)\Delta\overline{\alpha}_i}{\left(\gamma\tau_0 + \alpha_i^o(0) + t\Delta\overline{\alpha}_i\right)} \tag{56}$$

$$\text{APY}_i = \frac{1}{\alpha_i^o(0)}\sum_t \frac{(\alpha_i^o(0) + t\Delta\overline{\alpha}_i)(0.41)\Delta\overline{\alpha}_i}{\left(\gamma\tau_0 + \alpha_i^o(0) + t\Delta\overline{\alpha}_i\right)} \tag{57}$$

If we define

$$D_i(t) := \left(\gamma\tau_0 + \alpha_i^o(0) + t\Delta\overline{\alpha}_i\right) \tag{58}$$

then (56)-(57) can be written more simply as

$$\text{APY}_i^{root} = \sum_t \frac{\gamma(0.41)\Delta\overline{\alpha}_i}{D_i(t)} \tag{59}$$

$$\text{APY}_i = \sum_t \left(1 + \frac{t\Delta\overline{\alpha}_i}{\alpha_i^o(0)}\right)\frac{(0.41)\Delta\overline{\alpha}_i}{D_i(t)} \tag{60}$$

Now, to get the *entire* root APY, we would need to sum these rewards over all subnets, as the root stakers get dividends from each subnet:

$$\text{APY}^{root} = \sum_i\sum_t \frac{\gamma(0.41)\Delta\overline{\alpha}_i}{D_i(t)} \tag{61}$$

Similarly, we could compute the *average* subnet APY by summing (57) over all subnets and dividing by the number of subnets (let's call it N):

$$\text{APY}^{avg} = \frac{1}{N}\sum_i\sum_t \left(1 + \frac{t\Delta\overline{\alpha}_i}{\alpha_i^o(0)}\right)\frac{(0.41)\Delta\overline{\alpha}_i}{D_i(t)} \tag{62}$$

We next suggest the following goal; let us choose the tao weight such that the root APY is less than or equal to the average subnet APY:

$$\text{APY}^{root} \leq \text{APY}^{avg} \tag{63}$$

Explicitly, this becomes

$$\sum_i\sum_t \frac{\gamma(0.41)\Delta\overline{\alpha}_i}{D_i(t)} \leq \frac{1}{N}\sum_i\sum_t \left(1 + \frac{t\Delta\overline{\alpha}_i}{\alpha_i^o(0)}\right)\frac{(0.41)\Delta\overline{\alpha}_i}{D_i(t)} \tag{64}$$

This can be rearranged into the following:

$$0 \leq \frac{1}{N}\sum_i\sum_t \left(1 - \gamma N + \frac{t\Delta\overline{\alpha}_i}{\alpha_i^o(0)}\right)\frac{(0.41)\Delta\overline{\alpha}_i}{D_i(t)} \tag{65}$$

We note that nearly ever quantity in (65) is positive, except for the quantity $1 - \gamma N$. One way to guarantee that inequality (65) holds is to demand that $0 \leq 1 - \gamma N$. But this just simplifies to the following condition:

$$\gamma \leq 1/N \tag{66}$$

Thus, while this is surely a conservative choice, we can at least say the following:

> *If we take the tao weight to be the reciprocal of the number of subnets, then the root APY will be less than or equal to the average subnet APY.*

This could be a useful heuristic for choosing the tao weight, or at least gauge the appropriate order of magnitude.

Interestingly, we can use expressions (60) and (61) to get an approximate closed form expression for the APY values. Specifically, if we replace the discrete sums with integrals (which is reasonable given how quickly the blocks happen), then we find the following:

$$\begin{aligned}\text{APY}^{root} &= \sum\nolimits_i \sum_t \frac{\gamma(0.41)\Delta\overline{\alpha}_i}{\left(\gamma\tau_0 + \alpha_i^o(0) + t\Delta\overline{\alpha}_i\right)} \\ &\approx \sum\nolimits_i \int \frac{\gamma(0.41)\Delta\overline{\alpha}_i}{\left(\gamma\tau_0 + \alpha_i^o(0) + t\Delta\overline{\alpha}_i\right)} dt \\ &= \sum\nolimits_i \gamma(0.41)\ln\left(\gamma\tau_0 + \alpha_i^o(0) + t\Delta\overline{\alpha}_i\right)\end{aligned} \tag{67}$$

$$\begin{aligned}\text{APY}_i &= \sum_t \left(1 + \frac{t\Delta\overline{\alpha}_i}{\alpha_i^o(0)}\right)\frac{(0.41)\Delta\overline{\alpha}_i}{\left(\gamma\tau_0 + \alpha_i^o(0) + t\Delta\overline{\alpha}_i\right)} \\ &\approx \int \left(1 + \frac{t\Delta\overline{\alpha}_i}{\alpha_i^o(0)}\right)\frac{(0.41)\Delta\overline{\alpha}_i}{\left(\gamma\tau_0 + \alpha_i^o(0) + t\Delta\overline{\alpha}_i\right)} dt \\ &= (0.41)\left(\frac{\Delta\overline{\alpha}_i}{\alpha_i^o(0)}t - \frac{\gamma\tau_0}{\alpha_i^o(0)}\ln\left(\gamma\tau_0 + \alpha_i^o(0) + t\Delta\overline{\alpha}_i\right)\right)\end{aligned} \tag{68}$$

Thus, we see that the subnet token holders have returns that grow **linearly** over time, whereas the passive root stakers have returns that only grow **logarithmically**.

Section 5.2: The Mechanics of Halving

In this section, we look at the math behind the halving schedule as computed in practice. We consider a scenario where we have an initial quantity (say, 1) that we add up N many times. After this point, we halve the quantity and add it up N many times again. This is done indefinitely:

$$\big[1+1+...+1\big]+\big[\tfrac{1}{2}+\tfrac{1}{2}+...+\tfrac{1}{2}\big]+\big[\tfrac{1}{4}+\tfrac{1}{4}+...+\tfrac{1}{4}\big]+...$$

This is meant to represent the accumulating token supply. For example, if we let $N = 10,500,000$, then we have the TAO halving schedule represented as the following:

$$\big[10,500,000\big]+\big[5,250,000\big]+\big[2,625,000\big]+... \tag{69}$$

Now, let us denote the current accumulated supply by S. The *eventual total* token supply, which we denote by S^*, will be given by using the well known formula for summing a convergent geometric series:

$$\big[N\times 1\big]+\big[N\times\tfrac{1}{2}\big]+\big[N\times\tfrac{1}{4}\big]+...=N\Big[\tfrac{1}{1-1/2}\Big]=2N$$

and so we see that

$$S^*=2N \tag{70}$$

Thus, for example, if $N = 10,500,000$, then we see that the total supply of TAO will approach $S^* = 21,000,000$.

Now, suppose we are currently at some point in this process, where the amounts being currently summed are $(1/2)^k$, indicated below:

$$...+\big(N\times\tfrac{1}{2^k}\big)+... \tag{71}$$

Let $S_\downarrow$ denote the accumulated supply up to the previous halving point, which we can sum as a *finite* geometric series:

$$S_\downarrow = K\times\sum_{n=0}^{k\text{-}1}(1/2)^n = N\left(\frac{1\text{-}(1/2)^k}{1\text{-}(1/2)}\right)=2N\big(1\text{-}(1/2)^k\big) \tag{72}$$

In a similar way, we let $S_\uparrow$ denote the future accumulated supply at the point of the next future halving event:

$$S_\uparrow = S^*\big(1\text{-}(1/2)^{k+1}\big) \tag{73}$$

We can solve equation (73) for the exponent k:

$$k=-\log_2\left(1-\frac{S_\downarrow}{S^*}\right) \tag{74}$$

We can do the same for equation (74) and solve for $k+1$:

$$k+1=-\log_2\left(1-\frac{S_\uparrow}{S^*}\right) \tag{75}$$

Now, because the current supply satisfies $S_\downarrow \le S \le S_\uparrow$, and because the logarithm is a monotonically increasing function, then we can say the following:

$$k\le-\log_2\left(1-\frac{S}{S^*}\right)\le k+1 \tag{76}$$

In other words, we can compute k by the expression

$$k=\left\lfloor-\log_2\left(1-\frac{S}{S^*}\right)\right\rfloor \tag{77}$$

Thus, at any point in time, we use the cumulative supply S and the total target supply S^*, and we can compute the amount of block emission $N/2^k$ according to (77).

Section 5.3: Root reward claiming (accounting)

The root dividends (which are taken from the Alpha emissions) may be given to the root stakers in several ways. Initially, it will be done by auto-selling the dividends (in Alpha) into the subnet AMM and thus receiving TAO to then pass on to the root stakers (distributed on a pro-rate basis).

However, there may be a point when this is switched over to an alternative system where Alpha dividends are given directly to the root stakers *as Alpha tokens*. In this case, the Alpha dividends that are delivered to the root subnet accumulate and can be claimed at any point by users (again, on a pro-rata basis, depending on how much TAO they hold). In this section, we look at a particular kind of efficient accounting done for the root dividends. This is straightforward to compute and store in principle, but the efficient accounting method in use is somewhat opaque. To describe it, we define the following variables:

- $\Delta\alpha$ = dividend injection of Alpha
- τ = TAO staked on root by a user
- T = total TAO staked on root
- ρ = a variable called 'rewards_per_tao'
- δ = a variable called 'debt'
- α_c = Alpha that can be claimed by user

Now, for an injection $\Delta\alpha$, a user is entitled to the quantity:

$$\alpha_c=(\tau/T)\Delta\alpha \tag{78}$$

In other words, it's the fraction of the Alpha rewards that is equal by the user's fraction of TAO ownership. Next we define the following update formulas (where an apostrophe denotes an updated value):

$$\text{(initialize)}\quad \rho=0,\quad \delta=0 \tag{79}$$

$$\text{(at stake }\Delta\tau\text{)}\quad \big\{\tau'=\tau{+}\Delta\tau,\quad \delta'=\delta{+}\rho\Delta\tau\big\} \tag{80}$$

$$\text{(at inject }\Delta\alpha\text{)}\quad \rho'=\rho{+}\Delta\alpha/T \tag{81}$$

$$\text{(to claim)}\quad \alpha_c=\tau\rho-\delta \tag{82}$$

We can confirm by induction that these formulas (77)-(80) achieve their desired goals. First, after an initial injection, equation (80) reduces to (76). Then, if we assume formula (80) holds for our inductive step, we consider another stake and/or injection. The amount that user should now be able to claim is $\alpha_c{+}(\tau'/T)\Delta\alpha$. However, we can rearrange this:

$$\begin{aligned}\alpha_c' &= \alpha_c+(\tau'/T)\Delta\alpha\\ &=(\tau\rho-\delta)+(\tau{+}\Delta\tau)\Delta\alpha/T+\rho\Delta\tau-\rho\Delta\tau\\ &=(\tau{+}\Delta\tau)(\rho{+}\Delta\alpha/T)-(\delta{+}\rho\Delta\alpha)\\ &=\tau'\rho'-\delta'\end{aligned} \tag{83}$$

which completes the inductive step.