Appendix A
An Description of the Growth Rate

We introduce some commonly used terms and describe the growth rate as follows. Say, given an interval I_k, the interval's *absolute change* of hashes during a time period of length Δt is commonly defined as follows,

$$\int_{t_{k-1}+\Delta t}^{t_k+\Delta t} h_i(t)dt - \int_{t_{k-1}}^{t_k} h_i(t)dt \tag{1}$$

The *percent change* is the absolute change with respect to the current interval's hashes $\int_{t_{k-1}}^{t_k} h_i(t)dt$. It is commonly defined as follows,

$$\frac{\int_{t_{k-1}+\Delta t}^{t_k+\Delta t} h_i(t)dt - \int_{t_{k-1}}^{t_k} h_i(t)dt}{\int_{t_{k-1}}^{t_k} h_i(t)dt} \tag{2}$$

The *growth rate* is the percent change with respect to the time period of length Δt, which is used to describe the rate of the percent change. It is commonly defined as follows,

$$\frac{\int_{t_{k-1}+\Delta t}^{t_k+\Delta t} h_i(t)dt - \int_{t_{k-1}}^{t_k} h_i(t)dt}{\Delta t \int_{t_{k-1}}^{t_k} h_i(t)dt} \tag{3}$$

where the growth rate has two forms: the *continuous-time growth rate* and the *discrete-time growth rate*. Besides, the relationship between the continuous-time growth rate and the discrete-time growth rate is as follows: We can use the mathematical property of the continuous-time growth rate to calculate the hash rate function $\int_{t_{k-1}}^{t_k} h_i(t)dt$, from which the discrete-time growth rate with $\Delta t = |t_{k+1} - t_k|$ is simply available.

Appendix B
Estimate Daily Computing Power

We introduce the method of estimating computing power. People commonly use the concept of Bitcoin *mining difficulty* to estimate the hash rate of the Bitcoin network.

Let denote the mining difficulty of the Bitcoin network at time t by $\xi(t)$. Bitcoin uses $\xi(t)$ to roughly guarantee that the duration of each block takes 600 seconds on average. Vividly, Bitcoin uses the historical block times to predict future block times. Say, $\xi(t)$ changes every 2,016 blocks time. For each change, its value for the future 2,016 blocks is set, such that the rate for the historical 2,016 blocks equals 600 seconds per block. Hence, the relationship between $\xi(t)$ and the hash rate of the Bitcoin network at time t can be roughly predicted as follows,

$$\sum_{i=1}^{N} h_i(t) \approx \frac{2^{32}\xi(t)}{600 \text{ secs}} \tag{4}$$

where $t \in I_k$ and the length of I_k equals one day. However, mining difficulty has a *property* that after each change, $\xi(t)$ will remain a constant in a period of 2,016 blocks (i.e., around 2,016 $\times$ 600 seconds) until the next change. Such a property leads to a problem in Eq. (4), where it predicts that the hash rate of the Bitcoin network should remain a specific value in the next period of 2,016 $\times$ 600 seconds. Nevertheless, in real-world situations, the hash rate of the Bitcoin network may be different from that specific value.

Therefore, we need to measure the real-world hash-rate related data to adjust the Eq. (4). Based on the Bitcoin mining mechanism, we measure the number of blocks found by mining pools, and have a better estimation on the hash rate of the Bitcoin network at time t as follows,

$$\sum_{i=1}^{N} h_i(t) \approx \frac{\#_{\text{actual bitcoin}}}{\#_{\text{expected bitcoin}}} \times \frac{2^{32}\xi(t)}{600 \text{ secs}} \tag{5}$$

where $t \in I_k$ and the length of I_k equals one day. The symbol $\#_{\text{actual bitcoin}}$ means the actual number of blocks found on the Bitcoin network in the interval. And, the symbol $\#_{\text{expected bitcoin}}$ means the expected number of blocks to be found on the Bitcoin network in this interval (i.e., $\#_{\text{expected bitcoin}} = \left\lfloor \frac{t_k - t_{k-1}}{600 \text{ secs}} \right\rfloor$ blocks), where each block is mathematically expected to take exactly 600 seconds.

Similarly, the number of hashes of the Bitcoin network in the interval is as follows,

$$\sum_{i=1}^{N} \int_{t_{k-1}}^{t_k} h_i(t)dt \approx \frac{\#_{\text{actual bitcoin}}}{\#_{\text{expected bitcoin}}} \times \int_{t_{k-1}}^{t_k} \frac{2^{32}\xi(t)}{600 \text{ secs}} dt \tag{6}$$

From Eq. (5), the hash rate of mining pool i at time t can be estimated as follows,

$$h_i(t) \approx \frac{\#_{\text{actual pool } i}}{\#_{\text{expected bitcoin}}} \times \frac{2^{32}\xi(t)}{600 \text{ secs}} \tag{7}$$

where $t \in I_k$ and the length of I_k equals one day. The symbol $\#_{\text{actual pool } i}$ means the actual number of blocks found by mining pool i in the interval.

Similarly, the number of hashes of mining pool i in the interval is as follows,

$$\int_{t_{k-1}}^{t_k} h_i(t)dt \approx \frac{\#_{\text{actual pool } i}}{\#_{\text{expected bitcoin}}} \times \int_{t_{k-1}}^{t_k} \frac{2^{32}\xi(t)}{600 \text{ secs}} dt \tag{8}$$

where the definite integral $\int_{t_{k-1}}^{t_k} \xi(t)dt$ is simple since the value of $\xi(t)$ changes once every $20,160$ minutes. And the mining difficulty is a field of Bitcoin block header that is permanently stored in the Bitcoin blockchain. By default, the related parameters about daily computing power are estimated with a reference to Eqs. (5-8).

Appendix C
Proofs

C.1. Proof of Proposition 4.5

Proof: Let the number of hashes of new mining pools in the interval I_k be $H_j(I_k)$, where $j \notin [1, N]$ and $H_j(I_k) = 0$. In the next interval I_{k+1}, it will add $H_j(I_{k+1})$ hashes to the Bitcoin network, where $H_j(I_{k+1}) > 0$ since mining activities are profitable. Also we have $R(I_{k+1}) = R(I_k)$ and $\sum_{i=1}^N H_i(I_{k+1}) = \sum_{i=1}^N H_i(I_k)$ since $R(I_k)$, $\sum_{i=1}^N H_i(I_k)$ are stable. Therefore, the problem of adding $H_j(I_{k+1})$ hashes to the Bitcoin network for maximizing the expected payoff of new mining pools in the next interval I_{k+1} can be formulated as the optimization problem (9).

$$\begin{aligned} \underset{H_j(I_{k+1})}{\text{maximize}} \quad & \frac{H_j(I_{k+1})R(I_k)}{H_j(I_{k+1}) + \sum_{i=1}^N H_i(I_k)} - cH_j(I_{k+1}) \\ \text{subject to} \quad & \frac{R(I_k)}{\sum_{i=1}^N H_i(I_k)} > c \end{aligned} \tag{9}$$

It implies that the solutions of either increasing too much computing power or too little, to the Bitcoin network are not optimal. Therefore, we are going to find the optimal one. Specifically, we let the first derivative $\frac{\partial}{\partial H_j(I_{k+1})}\left[\frac{H_j(I_{k+1})R(I_k)}{H_j(I_{k+1})+\sum_{i=1}^N H_i(I_k)} - cH_j(I_{k+1})\right] = 0$, and get two saddle points $H_j(I_{k+1}) = -\sqrt{\frac{R(I_k)\sum_{i=1}^N H_i(I_k)}{c}} - \sum_{i=1}^N H_i(I_k)$, $H_j(I_{k+1}) = \sqrt{\frac{R(I_k)\sum_{i=1}^N H_i(I_k)}{c}} - \sum_{i=1}^N H_i(I_k)$, respectively. The first saddle point can be removed since it does not satisfy the condition of $H_j(I_{k+1}) > 0$. The second saddle point can be proven to be the optimal one since the second derivative $\frac{\partial^2}{\partial H_j(I_{k+1})\partial H_j(I_{k+1})}\left[\frac{H_j(I_{k+1})R(I_k)}{H_j(I_{k+1})+\sum_{i=1}^N H_i(I_k)} - cH_j(I_{k+1})\right] = -2\sqrt{cR(I_k)\sum_{i=1}^N H_i(I_k)} < 0$. ■

C.2. Proof of Proposition 4.6

Proof: Let the number of hashes of an existing mining pool i in the interval I_k be $H_i(I_k)$, where $i \in [1, N]$, $H_i(I_k) > 0$. Mining pool i will increase/decrease $H_{i'}(I_{k+1})$ hashes to/from the Bitcoin network in the next interval I_{k+1}. Also, $R(I_{k+1}) = R(I_k)$, $\sum_{i=1}^N H_i(I_{k+1}) = \sum_{i=1}^N H_i(I_k)$ since $R(I_k)$, $\sum_{i=1}^N H_i(I_k)$ are stable. Moreover, there are no new mining pools. Therefore, the problem of increasing/decreasing $H_{i'}(I_{k+1})$ hashes to/from the Bitcoin network for maximizing the expected payoff of mining pool i in the next interval can be formulated as the optimization problem (10).

$$\begin{aligned} \underset{H_{i'}(I_{k+1})}{\text{maximize}} \quad & [H_{i'}(I_{k+1}) + H_i(I_k)] \times \\ & \left[\frac{R(I_k)}{H_{i'}(I_{k+1}) + \sum_{i=1}^N H_i(I_k)} - c\right] \\ \text{subject to} \quad & \frac{R(I_k)}{\sum_{i=1}^N H_i(I_k)} > c \end{aligned} \tag{10}$$

We are going to find the optimal solution. And similar to the previous optimization problem (9), the optimal solution is $\sqrt{\frac{R(I_k)\left(-H_i(I_k)+\sum_{i=1}^N H_i(I_k)\right)}{c}} - \sum_{i=1}^N H_i(I_k)$. It means that for an existing mining pool i whose hashes satisfy $H_i(I_k) < \left(1 - \frac{c\sum_{i=1}^N H_i(I_k)}{R(I_k)}\right)\sum_{i=1}^N H_i(I_k)$, the optimal strategy is to add $\sqrt{\frac{R(I_k)\left(-H_i(I_k)+\sum_{i=1}^N H_i(I_k)\right)}{c}} - \sum_{i=1}^N H_i(I_k)$ hashes to the Bitcoin network in the next interval. Otherwise, the optimal solution is to decrease $\sum_{i=1}^N H_i(I_k) - \sqrt{\frac{R(I_k)\left(-H_i(I_k)+\sum_{i=1}^N H_i(I_k)\right)}{c}}$ hashes from the Bitcoin network in the next interval. ■

C.3. Proof of Proposition 4.8

Proof: Let the number of hashes of an existing mining pool i in the interval I_k be $H_i(I_k)$, where $i \in [1, N]$, $H_i(I_k) > 0$. Mining pool i will decrease $H_{i'}(I_{k+1})$ hashes from the Bitcoin network in the next interval I_{k+1} where $H_{i'}(I_{k+1}) > 0$ since mining activities are non-profitable. Also, $R(I_{k+1}) = R(I_k)$ and $\sum_{i=1}^N H_i(I_{k+1}) = \sum_{i=1}^N H_i(I_k)$ since $R(I_k)$, $\sum_{i=1}^N H_i(I_k)$ are stable. Therefore, the problem of decreasing $H_{i'}(I_{k+1})$ hashes from the Bitcoin network for maximizing the expected payoff of mining pool i in the next interval can be formulated as the optimization problem (11).

$$\begin{aligned} \underset{H_{i'}(I_{k+1})}{\text{maximize}} \quad & [-H_{i'}(I_{k+1}) + H_i(I_k)] \times \\ & \left[\frac{R(I_k)}{-H_{i'}(I_{k+1}) + \sum_{i=1}^N H_i(I_k)} - c\right] \\ \text{subject to} \quad & \frac{R(t)}{\sum_{i=1}^N H_i(I_k)} \leq c \end{aligned} \tag{11}$$

We are going to find the optimal solution. There are two cases.

Case 1. If an existing mining pool i whose hashes satisfy $\frac{R(I_k)}{-H_i(I_k)+\sum_{i=1}^N H_i(I_k)} > c$, then the optimization problem (11) can be represented the same form as the optimization problem (9). Therefore, after applying the proposition 4.5, we get the optimal solution $H_{i'}(I_{k+1}) = \sum_{i=1}^N H_i(I_k) - \sqrt{\frac{R(I_k)\left(-H_i(I_k)+\sum_{i=1}^N H_i(I_k)\right)}{c}}$ where $H_{i'}(I_{k+1}) > 0$.

Case 2. If an existing mining pool i whose hashes satisfy $\frac{R(I_k)}{-H_i(I_k)+\sum_{i=1}^N H_i(I_k)} \leq c$, then intuitively the optimal solution is to decrease all hashes $H_{i'}(I_{k+1})$ where $H_{i'}(I_{k+1}) = H_i(I_k)$ from the Bitcoin network in the next interval. ■

C.4. Proof of Theorem 4.9

Proof: Given a stable mining revenue R. The number of hashes of the Bitcoin network is $\sum_{i=1}^N H_i(I_k)$ hashes in interval I_k. We are going to prove that $\lim_{n\to\infty}\sum_{i=1}^N H_i(I_{k+n})$ will converge to a Nash equilibrium point among mining pools. There are two cases.

Case 1. If mining activities are profitable, then we apply the proposition 4.5, and know that extra hashes $\sqrt{\frac{R\sum_{i=1}^N H_i(I_k)}{c}} - \sum_{i=1}^N H_i(I_k)$ will be added to the Bitcoin network by new mining pools. It means that the number of hashes of the Bitcoin network in the next interval will be $\sum_{i=1}^N H_i(I_{k+1}) = \sum_{i=1}^N H_i(I_k) +$

$\left(\sqrt{\frac{R\sum_{i=1}^{N} H_i(I_k)}{c}} - \sum_{i=1}^{N} H_i(I_k)\right) = \sqrt{\frac{R\sum_{i=1}^{N} H_i(I_k)}{c}}$ hashes. So on and so forth, after n intervals, we have $\lim_{n\to\infty} \sum_{i=1}^{N} H_i(I_{k+n}) = \frac{R}{c}$.

Case 2. If mining activities are non-profitable, then we repeatedly apply the proposition 4.8, and know that the optimal solution for a major mining pool i to maximize its expected payoff is to decrease $\sum_{i=1}^{N} H_i(I_k) - \sqrt{\frac{(-H_i(I_k)+\sum_{i=1}^{N} H_i(I_k))R}{c}} > 0$ hashes from the Bitcoin network in the next interval, and finally go to *Case 1* of this proof. ■