Incentive Compatibility: Given the simple state space of RPPLNS, we provide a recursion which allows and the fraction of shares controlled by to the right we show the optimal strategy for using dynamic programming and compared against expected the honest mining reward. In the figures we can recursively define the gain of a strategic miner after \( k \) steps as \( q_k(ℓ, s, b) \). This can be computed using dynamic programming and compared against expected the honest mining reward. In the figures to the right we show the optimal strategy for \( m_1 \) given \( α \), the hash power of the honest in-pool miners and the fraction of shares controlled by \( m_1 \).

Notice that as long as the fraction of shares controlled by \( m_2 \) is not too far from his hash power, the result is honest behaviour. For the case where his hash power is either too high or the fraction of shares he controls too different, strategic behaviour (which also happens for PPLNS and can be explained theoretically) does occur. These situations are unlikely to happen in practice.

### Properties of RPPLNS:

Suppose that \( m_1 \) is honest with hash power \( α \).

1. Fairness: their expected per-turn block reward is \( \frac{1}{2} \) in an RPPLNS mining pool.
2. Variance Reduction: the variance of the per-turn block reward is \( \frac{1}{2} (α - α^2) + \frac{1}{4} α \).
3. Robustness to pool hopping: Over a finite time horizon, the payoff \( m_1 \) obtains is only a function of the amount of time they dedicate to mining for the pool.
4. Incentive Compatibility: given the simple state space of RPPLNS, we provide a recursion which allows us to empirically compute hash rates and bag distributions where honest mining dominates an arbitrary strategic deviation.

### Incentive Compatibility

One advantage of RPPLNS over PPLNS is that the state from the perspective of the strategic miner is just a tuple \((ℓ, s, b)\), where \( ℓ \) is the number of published shares, \( s \) is the number of private shares and \( b \) the number of private blocks. In PPLNS, \( ℓ \) would have to be the whole queue.

We can recursively define the gain of a strategic miner after \( k \) steps as \( q_k(ℓ, s, b) \). This can be computed using dynamic programming and compared against expected the honest mining reward. In the figures to the right we show the optimal strategy for \( m_1 \) given \( α \), the hash power of the honest in-pool miners and the fraction of shares controlled by \( m_1 \).

Notice that as long as the fraction of shares controlled by \( m_2 \) is not too far from his hash power, the result is honest behaviour. For the case where his hash power is either too high or the fraction of shares he controls too different, strategic behaviour (which also happens for PPLNS and can be explained theoretically) does occur. These situations are unlikely to happen in practice.