High Level Goals

- Means to reduce network-wide orphan/DOA rate would allow the pool to grow larger as fewer miners would quit in frustration.
- Means to reduce variance would also help to avoid miners quitting.
- Means to reduce the frequency of dust payouts makes it less costly to spend p2pool outputs, making p2pool friendlier to miners.

Brain Storming Ideas

1. **“Fee on each share” idea - promotes growth**
   Charge a tiny fee (which gets distributed evenly over the P2Pool) on each share mined, resulting in large miners naturally wanting to increase their share difficulty to concentrate their work in fewer shares. Allows smaller miners on by decreasing share difficulty, though smaller miners have to pay slightly more than large miners for the privilege (unless they’re fine with high variance).
   a. Criteria miners might use for choosing their share difficulty:
      i. Start with difficulty share that would result in one share every half hour
      ii. Increase difficulty (decreasing effective fee) until fee <= 10%
   b. Parameters would be tweakable by users
   c. Allows users to choose their own tradeoff between variance and fees by specifying maximum fee or maximum time between shares

2. **“Multiple addresses on each share” idea - promotes growth**
   Allow each P2Pool share to pay to multiple addresses, resulting in public P2Pool nodes having a much greater benefit by allowing users to work together for shares. Allows smaller miners on by decreasing apparent share difficulty.
   a. In a way similar to (1), a small fee would be levied on multi-address shares to prevent spam.

3. **“Trustless Accumulator - Store payouts owed in sharechain” idea - promotes growth & decreases dust**
   a. Payouts smaller than a (potentially user-tweakable) dust threshold are stored the sharechain and the money that would go to those payouts is instead used to pay owed stored payouts, largest to smallest
   b. In order to bootstrap this sharechain debt, the extra money is donated to other miners during the initial stage where paying out stored payouts would happen before they’ve accumulated enough to not qualify as dust

   c. In order to keep the “payouts owed” liquid, potentially implement something like SMPPS that uses the “payouts owed” buffer

4. **“(Semi-)Trustless accumulator” idea - promotes growth & decreases dust**
   a. Only current known way to implement this:
      i. N semi-trusted nodes are determined, including some chosen randomly

        (probably weighted by hash rate) and probably some hardcoded (for example, my node)
    ii. M chosen arbitrarily, e.g. 75% + 1 hardcoded node
    iii. Blocks distribute some and pay extra towards current multisig M of N destination
    iv. Payment to multisig destination would also include hash covering description of who is owed what
        1. Description of who is owed what is permanently archived with every p2pool node
    v. Semi-trusted nodes periodically combine payments to new multisig destination based on newly chosen semi-trusted nodes and pay addresses exceeding threshold
  b. Complex
  c. Simple approach doesn't completely guarantee that accumulator can not be stolen (if it's too easily accessible, and the M nodes are compromised) or lost (if it's hard to access and N-M+1 nodes go offline)
    i. Adding a hardcoded node that is required to confirm multisig transactions prevents stealing (as long as hardcoded node owner (e.g. Forrest) is trusted not to collude with M other randomly chosen nodes)
    ii. Something can probably be done (reasonably sure, more research needed) with nLockTime to release payouts if lots of time passes without anything happening, which prevents accumulator from being lost

5. **"Create a standard P2Pool sub-pool" idea - promotes growth & decreases dust**
  a. Created implementation would replace ad-hoc developed solutions currently present
  b. Could be made easily auditable
  c. Consider AGPL? Would make pool op cheating illegal in another way

6. **"Multiple chains" idea - promotes growth**
   Have multiple parallel chains that work somewhat independently, but all are paid out together in blocks. Allows smaller miners on by decreasing share difficulty.
  a. Old idea, somewhat complex and groundbreaking
  b. Trustless accumulator would allow for lower difficulty alt chain without dusty payouts.

7. **Reduce Share Propagation Latency**
   Currently the more peers or miners you have, the longer it takes for a p2pool node to sequentially send all peers a new share before it restarts for new work. Anything that can be done to reduce the share propagation latency or allow new work to restart sooner can reduce the network-wide share orphan and DOA rate, which would allow the p2pool network to grow larger.
  a. Multiple threads possible? Upon incoming share, validate once then send

outgoing shares in a separate thread. This would allow new work to commence sooner.

8. **Per-Peer Statistical Tracking**
    a. Keep track of the usefulness of peers. If a peer sent you the least quantity of new shares and it also fails to respond to ping in a timely manner then disconnect it. Add some randomness to the peer disconnection choice algorithm and allow a long period of time for a new peer to prove itself. Over time this would naturally reorganize the p2pool network to have poorly bandwidth resourced nodes pushed to the edges. The high bandwidth nodes would naturally become highly connected to each other and network-wide orphan rate would decrease.
    b. Are peer share sending currently blocking in any way? The previous multi-threaded network handling might help.

9. **Chain-node-less P2Pool nodes**
    a. Using Bitcoin's UTXO set idea and currently possible proofs of facts about the chain, it should be possible for the P2Pool network to consist of a mix of nodes that have Bitcoin/Litecoin nodes and ones that work alone.
    b. This would let P2Pool run on much lower-powered machines.