Stateless mining is important for decentralization. Below are some strategies for stateless mining.

(1) Mine empty blocks.

(2) 21000 gas balance transfers. Take a transaction, recover the sender and receiver addresses, somehow obtain their witnesses (sender witness may be part of the transaction), and maintain a transaction pool which updates sender and receiver witnesses.

(3) Heuristics. Generalize (2) to contract call transactions by using heuristics to try to recover access lists. Unfortunately, the access list for an arbitrary transaction is undecidable a priori

. But maybe some transactions allow a fast procedure to find their access lists – an open question.

(4) Promised access list. A transaction can be accompanied with a second transaction which claims that the first transaction honors a specific access list. If the first transaction violates this access list, then the second transaction can be used to somehow penalize the sender.

The remaining strategies require protocol changes.

(5) Protocol-level access lists. Add a transaction field for an optional but binding access list. Like (4), but with semantics to check for access list violations. The reason that this list is “optional” is to support transactions which have undecidability or slow/awkward decidablility of their access lists. There is an optimization. Warning: access lists allow 51% attack censorship black-lists.

(6) Stateless-mining-friendly syntax. Hard-fork to new syntax/semantics rules which are not Turing complete, but allow quickly deciding the access list, even for adverserial inputs. Maybe this abstract machine would be a finite state machine like described here and independently prototyped here, or motivated by something like Vyper. Not sure how this would look – it may end up being similar to just declaring the access list like in (5).

Any stateless mining strategies missing? Any feedback on the above strategies?