4.5 Transaction types

This document is to serve as a supplement to the main manuscript titled "Trustless parallel local search for effective distributed algorithm discovery", currently undergoing peer review at IEEE Blockchain: From Technology to Marketplaces.

In the following subsections we describe in detail every type of transaction with its arguments, effect, and validity.

4.5.1 submit_agent($UUID$, $stake$)

Sent by a searcher to notify that it has an agent that it wants to verify in the next tournament. Transaction arguments include an $UUID$ by which this agent will be identified in future communication and a *stake*, which is the amount of funds the node is willing to "stake" (or lock) on the agent. Staked tokens demonstrate the node's confidence in the agent. Stake can be retrieved by declaring the agent as unavailable (sending this transaction with a *stake* of 0).

The transaction is invalid if $stake < 0$, the searcher does not have enough tokens to stake, or the proposed $UUID$ has already been used for another agent, even if that agent has later been declared as unavailable.

4.5.2 publish_dataset($inputsURL$, $inputsHash$, $encryptedSignalsURL$, $signalsHash$)

Only exists in dataset domains. It is used by challengers to publish the inputs of their personal dataset (accessible through $inputsURL$) so that competing searchers can prove their agents' performance on it. The challenger generates a random AES-256 key and encrypts and uploads the correct dataset outputs (through $encryptedSignalsURL$). Hashes of the inputs and decrypted outputs are provided for future verification.

The transaction is invalid if it is not signed by a selected tournament challenger, if the same challenger already submitted a transaction of this type during the same tournament, or if the challenger submission deadline has passed.

4.5.3 submit_signal($agentUUID$, $encryptedSignal$)

Sent by a searcher to submit an AES-256 *encryptedSignal* from a specific agent ($agentUUID$). Every signal must be encrypted with a different, unique, AES key.

The transaction is valid if $agentUUID$ exists and is signed by a searcher that previously sent a valid submit_agent transaction for the same agent before tournament start.

Additionally, if the problem is real-time, the transaction is valid only if received on a specific real-time tick during the tournament. It is invalid if the same transaction type has already been sent by the same searcher and with the same $agentUUID$ for the same tick. If valid signal transactions for all ticks throughout a tournament are not received, the searcher is disqualified.

If the problem is *dataset*, the transaction is valid only if received during an active tournament. It is invalid if the same transaction type has already been sent for the same $agentUUID$. If not received, the searcher is disqualified from this tournament.

4.5.4 publish_dataset_decryption_key(key)

Only exists in dataset domains. Every tournament challenger has to send this after the end of a tournament and reveal the encryption key that was used to encrypt the correct dataset outputs in publish_dataset.

The transaction is invalid if not signed by a non-disqualified tournament challenger, not received at the tournament end, or the same transaction type has already been sent by the same challenger after the end of the same tournament.

If a tournament challenger does not submit this valid transaction, they are disqualified.

4.5.5 publish_signal_decryption_key(*agentUUID*, *key*)

Sent by a searcher a specific amount of time after signing submit_signal to reveal the AES-256 key by which the original signal was encrypted.

The transaction is valid if *agentUUID* is a previously registered agent by the signing non-disqualified searcher and *key* has not been previously used in this tournament.

If the problem is real-time, the transaction is only valid if received at a specific real-time tick and *key* successfully decrypts the signal received in submit_signal transaction for *agentUUID* from the previous real-time tick.

If the problem is dataset, the transaction is only valid if received within a $timeTolerance$ of the tournament end and *key* decrypts the previously submitted signal.

If the signal is for an agent competing in an active tournament and a valid key is not received by the end deadline, the searcher which owns the agent is disqualified.

4.5.6 publish_tournament_ranking(*ranking*)

Submits the local tournament ranking of one validator.

Transaction is only valid if:

- The signer is a validator;
- The transaction is received after a tournament end and before the ranking deadline;
- The tournament is successful;
- This validator has not already submitted a ranking for this tournament.

4.5.7 publish_agent_price(*agentUUID*, *scheme*, *price*, *stake*)

Advertises purchasing rules for a previously verified agent.

A searcher can allow their agent to be rented, or subscribed to, by other nodes. There are two types of payment schemes - paying the *price* for every time you use an agent, subscribing to the agent and being able to use it for a specific period, or directly buying the algorithm. This transaction locks *stake* amount of tokens from the searcher's account. If $stake = 0$, this action will declare the agent as unavailable, and return the previously held *stake* back to the searcher.

The transaction is valid if the signer has successfully had *agentUUID* validated in a tournament, if the signer does not have balance for the stake, and if the agent has not been made unavailable.

4.5.8 **publish_data_price**(*dataUUID*, *dataParams*, *scheme*, *price*, *stake*)

Advertises purchasing rules for a dataset.

A data node can freely publicize what data it intends to sell to searchers for the training of their agents. Details, including the shape, frequency, and further description of the data features are included in *dataParams*, which is a domain-specific data structure. The *scheme*, *price*, and *stake* parameters have the same functionality as in publish_agent_price.

The transaction is valid if the signer has enough balance to lock the stake, *dataUUID* has not been previously assigned to an agent or data provider, and *dataUUID* has not been made unavailable previously.

4.5.9 **rent**(*UUID*, *quantity*)

Purchases access to a previously advertised agent or data.

Any node can rent a published agent or data from a data node. Signing this transaction will withdraw $quantity * price + rentFee$ domain tokens from the sender, where *price* is the specific cost of the agent or data to which *UUID* refers. The node that created the agent or data receives $quantity * price$ tokens. The *rentFee* cost is withdrawn to restrict potential transaction spam, and is sent to the next tournament reward. Afterwards, the buying node has to establish off-chain contact with the receiving node and agree upon a delivery method, which is domain-specific.

The transaction is valid if $quantity \geq 1$, the signer has enough balance to pay for the fees, *UUID* exists and has not been made unavailable, and a price for it has already been published through publish_agent_price or publish_data_price.

If *UUID* refers to an agent whose payment scheme is to sell the algorithm directly, the transaction is invalid if $quantity \neq 1$.