RL-based Blockchain Optimization — Flowchart → Algorithm Mapping

Purpose (one line)

Explain, in plain terms, which algorithm/module performs each box in the flowchart so a non-technical reviewer understands who does what.

Explain, in plain terms, which algorithm/module performs each box in the flowchart so a non-technical reviewer understands who does what.

Mapping (flowchart box → what performs it & what it does)

1. Collect Local Metrics

Performed by: Simulator / node sensors.

Does: Measures per-node values (peer latency, mempool depth, available bandwidth) that form the agent’s input.

2. RL Policy Decision (Agent observes state vector)

Performed by: Policy network (DDPG or P-DQN).

Does: Looks at the measured state and decides the next action (peer choices, block size, send timing).

3. Policy Network Outputs

Performed by: Policy output stage.

Does: Converts the decision into concrete parameters (neighbor weights or chosen strategy, block assembly settings).

4. Select Neighbor Set for Gossip Protocol

Performed by: Action post-processing.

Does: Picks which peers to contact based on the policy output.

5. Block Assembly Parameters (size, selection priority)

Performed by: Action post-processing.

Does: Sets block size and which transactions to include first.

6. EXECUTE ACTIONS — Peer Sampling & Transaction Propagation / Assemble Block & Propagate

Performed by: Simulator (SimPy).

Does: Applies the chosen gossip and block actions and simulates message/block propagation across the network.

7. MEASURE OUTCOMES (latency, throughput, orphan rate)

Performed by: Simulator + metrics logger.

Does: Records performance after actions (confirmation time, transactions/sec, orphaned blocks).

8. COMPUTE REWARD

Performed by: Reward module.

Does: Converts measured outcomes into a single score (higher reward = better performance) using weighted latency/throughput/orphan terms.

9. UPDATE RL POLICY (store transition → learn)

Performed by: Prioritized Experience Replay (PER) + learning algorithm.

Does: Stores experiences, prioritizes informative ones, and updates the policy:

- DDPG: learns continuous controls (per-peer weights, continuous block params).
- P-DQN: learns discrete strategy choices plus continuous parameters for each choice.

Short legend (one line each)

- SimPy (Simulator): runs the network and produces/records metrics.
- DDPG: learns continuous actions (fine-grained control).
- P-DQN: learns mixed discrete + continuous actions (strategy + params).
- PER (Prioritized Replay): stores experiences and focuses learning on the most useful ones.
- Reward module: turns latency/throughput/orphan metrics into the scalar learning signal.

One-sentence summary (for report/email)

A simulated node measures local network state, an RL policy (DDPG or P-DQN) decides gossip and block parameters, the simulator applies those actions and measures outcomes, and prioritized replay plus the chosen RL algorithm use those results to improve future decisions.