The Q-Whale Algorithm integrates Q-Learning, a reinforcement learning method, with the Whale Optimization Algorithm (WOA), a nature-inspired optimization technique. Initially, the Q-Table, which stores the value of state-action pairs, is initialized, and a population of whales (agents) is randomly generated in the solution space. During the exploration phase, each whale's state (its position) is evaluated, and actions (moves) are selected using an epsilon-greedy policy to balance exploration and exploitation. After performing an action, the reward, typically the fitness of the new position, is calculated, and the Q-Table is updated based on the Q-Learning update rule. This updated Q-Table guides the whales' movements, enhancing their search efficiency. In the optimization phase, WOA mechanisms, such as the bubble-net hunting strategy, are used to update the whales' positions, but now these updates are informed by the Q-Table, directing the search towards more promising regions. This combined approach continues until a convergence criterion is met, resulting in an optimized solution that leverages both the adaptive exploration of Q-Learning and the powerful search capabilities of WOA.

The Q-Whale Algorithm combines Q-Learning, a reinforcement learning technique, with the Whale Optimization Algorithm (WOA), a nature-inspired optimization method. The key idea is to use Q-Learning to guide the exploration phase of WOA, helping it to efficiently search the solution space and avoid local optima.

**Components**

1. **Q-Learning**:
   * **Q-Table**: A table used to store the value of state-action pairs, representing the expected utility of taking a certain action in a certain state.
   * **Exploration**: Q-Learning explores the solution space and updates the Q-Table based on the rewards obtained from different state-action pairs.
2. **Whale Optimization Algorithm (WOA)**:
   * **Search Agents**: Whales (or agents) that search the solution space.
   * **Exploration and Exploitation**: WOA mimics the bubble-net hunting strategy of humpback whales, balancing exploration (searching new areas) and exploitation (refining known good solutions).

**Integration of Q-Learning with WOA**

**Initialization**

1. **Initialize Q-Table**: Set all Q-values to zero or some initial values.
2. **Initialize Whales**: Generate initial population of whales (agents) randomly in the solution space.

**Exploration Phase with Q-Learning**

1. **State Representation**: Define states based on the problem domain (e.g., current position of whales in the solution space).
2. **Action Selection**: Use an epsilon-greedy policy to select actions. With probability epsilon, select a random action (exploration); otherwise, select the action with the highest Q-value (exploitation).
3. **Perform Action**: Move the whale according to the selected action.
4. **Receive Reward**: Calculate the reward based on the new state (e.g., fitness value of the new position).
5. **Update Q-Table**: Update the Q-value using the Q-Learning update rule:

Q(s,a)←Q(s,a)+α(r+γmax⁡a′Q(s′,a′)−Q(s,a))Q(s,a)←Q(s,a)+α(r+γa′max​Q(s′,a′)−Q(s,a))

where αα is the learning rate, γγ is the discount factor, rr is the reward, ss is the current state, s′s′ is the new state, aa is the current action, and a′a′ is the new action.

**Optimization Phase with WOA**

1. **Position Update**: Use WOA mechanisms to update the position of whales based on the encircling prey, bubble-net attacking method, and search for prey.
2. **Exploit Q-Table**: Guide the position updates using the Q-Table to leverage learned information, improving the search process by directing whales towards promising regions.

**Convergence Check**

1. **Termination Condition**: Check if the stopping criteria (e.g., maximum iterations, convergence threshold) are met.
2. **Solution Extraction**: Extract the best solution found by the whales.

**Pseudocode**

Initialize Q-Table with zeros

Initialize whale population randomly

For each episode:

For each whale (agent):

State = current position of whale

If random() < epsilon:

Action = random action (exploration)

Else:

Action = action with highest Q-value (exploitation)

Perform Action -> move whale

NewState = new position of whale

Reward = calculate reward based on NewState

Update Q-Table:

Q(State, Action) = Q(State, Action) + alpha \* (Reward + gamma \* max(Q(NewState, all actions)) - Q(State, Action))

State = NewState

Use WOA to update whale positions guided by Q-Table

If termination condition met:

Break

**Detailed Steps**

1. **Initialization**:
   * Initialize the Q-Table with zero values.
   * Generate an initial population of whales with random positions in the solution space.
2. **Q-Learning Exploration**:
   * For each whale, determine its current state based on its position.
   * Use epsilon-greedy policy to select an action (e.g., move to a new position).
   * Perform the selected action, moving the whale to a new position.
   * Calculate the reward for the new position (e.g., fitness value).
   * Update the Q-Table using the Q-Learning update rule.
3. **Whale Optimization**:
   * Use the WOA mechanisms to update the positions of whales.
   * Leverage the Q-Table to guide the updates, focusing the search on promising areas.
4. **Convergence**:
   * Check if the termination condition is met (e.g., maximum iterations, convergence).
   * If met, extract the best solution found by the whales.

**Applications**

The Q-Whale Algorithm can be applied to various optimization problems, including but not limited to:

* Function optimization
* Resource allocation
* Scheduling problems
* Machine learning hyperparameter tuning

By integrating Q-Learning with WOA, the algorithm can effectively balance exploration and exploitation, improving its ability to find optimal solutions in complex search spaces.