```
import numpy as np

grid_size = 5
start_state = (0, 0)
goal_state = (grid_size - 1, grid_size - 1)
obstacles = [(1, 1), (2, 2), (3, 1)]

actions = ['up', 'down', 'left', 'right']
num_actions = len(actions)

learning_rate = 0.1
discount_factor = 0.9
epsilon = 0.1
num_episodes = 1500

# Define Antworld environment
class Antworld:
    def __init__(self, grid_size, obstacles, start_state, goal_state):
        self.grid_size = grid_size
        self.obstacles = obstacles
        self.start_state = start_state
        self.goal_state = goal_state

    def reset(self):
        return self.start_state

    def step(self, state, action):
        if state == self.goal_state:
            return state, 0

        next_state = self.get_next_state(state, action)
        if next_state in self.obstacles:
            return state, -1
        elif next_state == self.goal_state:
            return next_state, 1
        else:
            return next_state, 0

    def get_next_state(self, state, action):
        x, y = state
        if action == 'up':
            return max(x - 1, 0), y
        elif action == 'down':
            return min(x + 1, self.grid_size - 1), y
        elif action == 'left':
            return x, max(y - 1, 0)
        elif action == 'right':
            return x, min(y + 1, self.grid_size - 1)

# Initialize Antworld environment
antworld_env = Antworld(grid_size, obstacles, start_state, goal_state)

Q_values = np.zeros((grid_size, grid_size, num_actions))

def select_action(state):
    if np.random.rand() < epsilon:
        return np.random.choice(actions)
    else:
        return actions[np.argmax(Q_values[state])]

def update_Q_value(state, action, reward, next_state, next_action):
    next_action_index = actions.index(next_action)
    Q_values[state][actions.index(action)] += learning_rate * (reward + discount_factor * Q_values[next_state][next_action_index] - Q_val

#SARSA Training
for episode in range(num_episodes):
    state = antworld_env.reset()
    action = select_action(state)
    while state != goal_state:
        next_state, reward = antworld_env.step(state, action) # corrected line
        next_action = select_action(next_state)
        update_Q_value(state, action, reward, next_state, next_action)
        state = next_state
        action = next_action

print("Learned Q - values:")
print(Q_values)

path = []

state = start_state
```

```
while state != goal_state:
    action = actions[np.argmax(Q_values[state])]
    path.append((state, action))
    state = antworld_env.get_next_state(state, action)
path.append((goal_state, None))

print("\nShortest Path:")
for step in path:
    print(step)
```

```
Learned Q - values:
[[[ 2.65339579e-01  2.31621820e-01  2.82591384e-01  4.00773064e-01]
  [ 3.73860920e-01 -6.13071881e-01  3.01308184e-01  4.51274829e-01]
  [ 4.38932594e-01  3.02270921e-01  3.29012314e-01  5.21699354e-01]
  [ 4.98550798e-01  5.76033887e-01  3.90948039e-01  4.09287120e-01]
  [ 3.27144907e-02  3.11729709e-02  4.91921198e-01  1.05200065e-01]]

 [[ 2.55714890e-01 -1.40022968e-04  2.74076633e-02 -4.01909711e-01]
  [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]
  [ 4.34168105e-01 -3.55910145e-01 -1.70776816e-01  1.06131761e-01]
  [ 4.58516674e-01  6.83628257e-01  3.68872346e-01  2.74711921e-01]
  [ 3.77917504e-01  5.75486648e-02  0.00000000e+00  0.00000000e+00]]

 [[-1.91532881e-06 -1.38541773e-03 -1.53929719e-03 -1.53900003e-02]
  [-1.90000000e-01 -2.71000000e-01 -3.90907273e-07 -1.90000000e-01]
  [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]
  [ 5.67332887e-01  7.79445638e-01 -3.08924380e-01  5.19550763e-01]
  [ 4.31263595e-03  9.97440572e-03  6.56887390e-01  3.24166889e-06]]

 [[-6.31024848e-07 -5.31441000e-06 -1.53900490e-02 -2.71000048e-01]
  [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]
  [-2.71000000e-01  7.17499980e-01 -4.41858557e-02  0.00000000e+00]
  [ 6.16327510e-01  8.77290211e-01  4.89696703e-01  3.34253384e-01]
  [ 5.25239367e-01  1.90000000e-01  7.75532840e-02  0.00000000e+00]]

 [[-7.29000000e-03 -7.29000000e-04 -7.44495697e-05 -1.01452087e-04]
  [-2.71000000e-01 -9.00000000e-03 -6.02365410e-04  1.60510093e-01]
  [ 2.74619583e-02  1.62196662e-01 -2.53114604e-03  8.97563144e-01]
  [ 7.46154943e-01  8.59184353e-01  7.05566409e-01  1.00000000e+00]
  [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]]

Shortest Path:
((0, 0), 'right')
((0, 1), 'right')
((0, 2), 'right')
((0, 3), 'down')
((1, 3), 'down')
((2, 3), 'down')
((3, 3), 'down')
((4, 3), 'right')
((4, 4), None)
```