Solving the Frozen Lake Problem with Value Iteration

Here's the Python code to solve the Frozen Lake problem using the value iteration algorithm:

```
import gym
import numpy as np

# Create the Frozen Lake environment
env = gym.make('FrozenLake-v1')

# Initialize the value function V(s) to zeros
V = np.zeros(env.observation_space.n)

# Set the discount factor
gamma = 0.9

# Set the convergence threshold
epsilon = 1e-6

# Value iteration algorithm
while True:
    delta = 0
    for s in range(env.observation_space.n):
        v = V[s]
        # Calculate the new value for state s using the Bellman equation
        V[s] = np.max([sum([p*(r + gamma*V[s_]) for p, s_, r, _ in env.P[s][a]]) for a in range(env.action_space.n)])
        delta = max(delta, np.abs(v - V[s]))
    if delta < epsilon:
        break

# Extract the optimal policy from the optimal value function
policy = np.array([np.argmax([sum([p*(r + gamma*V[s_]) for p, s_, r, _ in env.P[s][a]]) for a in range(env.action_space.n)]) for s in range(env.observation_space.n)])

# Display the optimal policy
print("Optimal policy:")
print(policy.reshape((4, 4)))

# Test the learned policy
total_reward = 0
num_episodes = 100
for _ in range(num_episodes):
    state = env.reset()
    done = False
```

```
    while not done:
        action = policy[state]
        state, reward, done, _ = env.step(action)
        total_reward += reward

# Calculate the average reward
average_reward = total_reward / num_episodes
print(f"Average reward over {num_episodes} episodes: {
    average_reward}")
```

This Python code uses the OpenAI Gym library to create the Frozen Lake environment and implements the value iteration algorithm to find the optimal policy. The algorithm iteratively updates the value function until convergence, then extracts the optimal policy from the optimal value function and tests it over multiple episodes to calculate the average reward.