Solving an Optimization Problem in an Active RIS-Assisted ISAC System Using DRL Algorithms (DDPG and SAC). ¶

This project focuses on solving an optimization problem in an active Reconfigurable Intelligent Surface (RIS)-assisted Integrated Sensing and Communication (ISAC) system using Deep Reinforcement Learning (DRL) algorithms. The goal is to maximize the dual-function radar sensing base station (DFRC BS) by jointly optimizing the beamforming matrix W W of the base station and the reflection phase shift matrix of the active RIS, all while ensuring the Signal-to-Interference-plus-Noise Ratio (SINR) of the communication user is maintained. The project employs DRL algorithms, specifically Deep Deterministic Policy Gradient (DDPG) and Soft Actor-Critic (SAC), to achieve the optimization. It involves creating a custom environment in OpenAI Gym, integrating it with Stable Baselines3 for training the DRL models, and comparing the performance and convergence of ordinary DDPG, double DDPG, and SAC algorithms through simulation and graphical analysis. The outcome of this project aims to enhance the efficiency and performance of ISAC systems in modern wireless communication networks.

Installing the necessary libraries

In [2]:
```
!pip install stable_baselines3 #Install if necessary or lacking
```

```
Collecting stable_baselines3
  Downloading stable_baselines3-2.3.2-py3-none-any.whl.metadata (5.1 kB)
Collecting gymnasium<0.30,>=0.28.1 (from stable_baselines3)
  Downloading gymnasium-0.29.1-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-
packages (from stable_baselines3) (1.26.4)
Requirement already satisfied: torch>=1.13 in /usr/local/lib/python3.10/dist-
packages (from stable_baselines3) (2.3.1+cu121)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-
packages (from stable_baselines3) (2.2.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packa
ges (from stable_baselines3) (2.1.4)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-p
ackages (from stable_baselines3) (3.7.1)
Requirement already satisfied: typing-extensions>=4.3.0 in /usr/local/lib/pyt
hon3.10/dist-packages (from gymnasium<0.30,>=0.28.1->stable_baselines3) (4.1
2.2)
Collecting farama-notifications>=0.0.1 (from gymnasium<0.30,>=0.28.1->stable_
baselines3)
  Downloading Farama_Notifications-0.0.4-py3-none-any.whl.metadata (558 byte
s)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-pac
kages (from torch>=1.13->stable_baselines3) (3.15.4)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packag
es (from torch>=1.13->stable_baselines3) (1.13.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-pac
kages (from torch>=1.13->stable_baselines3) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packa
ges (from torch>=1.13->stable_baselines3) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packa
ges (from torch>=1.13->stable_baselines3) (2024.6.1)
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch>=1.13->stable_baselin
es3)
  Using cached nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.wh
l.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch>=1.13->stable_basel
ines3)
  Using cached nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.w
hl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch>=1.13->stable_baselin
es3)
  Using cached nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.wh
l.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch>=1.13->stable_baselines3)
  Using cached nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl.meta
data (1.6 kB)
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch>=1.13->stable_baselines3)
  Using cached nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl.met
adata (1.5 kB)
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch>=1.13->stable_baselines3)
  Using cached nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl.met
adata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.2.106 (from torch>=1.13->stable_baselines
3)
  Using cached nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl.m
etadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.4.5.107 (from torch>=1.13->stable_baselin
```

```
es3)
  Using cached nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.wh
l.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.1.0.106 (from torch>=1.13->stable_baselin
es3)
  Using cached nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.wh
l.metadata (1.6 kB)
Collecting nvidia-nccl-cu12==2.20.5 (from torch>=1.13->stable_baselines3)
  Using cached nvidia_nccl_cu12-2.20.5-py3-none-manylinux2014_x86_64.whl.meta
data (1.8 kB)
Collecting nvidia-nvtx-cu12==12.1.105 (from torch>=1.13->stable_baselines3)
  Using cached nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metad
ata (1.7 kB)
Requirement already satisfied: triton==2.3.1 in /usr/local/lib/python3.10/dis
t-packages (from torch>=1.13->stable_baselines3) (2.3.1)
Collecting nvidia-nvjitlink-cu12 (from nvidia-cusolver-cu12==11.4.5.107->torc
h>=1.13->stable_baselines3)
  Downloading nvidia_nvjitlink_cu12-12.5.82-py3-none-manylinux2014_x86_64.wh
l.metadata (1.5 kB)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/
dist-packages (from matplotlib->stable_baselines3) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist
-packages (from matplotlib->stable_baselines3) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib->stable_baselines3) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib->stable_baselines3) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/d
ist-packages (from matplotlib->stable_baselines3) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dis
t-packages (from matplotlib->stable_baselines3) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/
dist-packages (from matplotlib->stable_baselines3) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python
3.10/dist-packages (from matplotlib->stable_baselines3) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist
-packages (from pandas->stable_baselines3) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/di
st-packages (from pandas->stable_baselines3) (2024.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-pac
kages (from python-dateutil>=2.7->matplotlib->stable_baselines3) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/d
ist-packages (from jinja2->torch>=1.13->stable_baselines3) (2.1.5)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.1
0/dist-packages (from sympy->torch>=1.13->stable_baselines3) (1.3.0)
Downloading stable_baselines3-2.3.2-py3-none-any.whl (182 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 182.3/182.3 kB 9.2 MB/s eta 0:00:
00
Downloading gymnasium-0.29.1-py3-none-any.whl (953 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 953.9/953.9 kB 35.2 MB/s eta 0:0
0:00
Using cached nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl (410.
6 MB)
Using cached nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl
(14.1 MB)
Using cached nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl
(23.7 MB)
```

```
Using cached nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl
(823 kB)
Using cached nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl (731.7
MB)
Using cached nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl (121.
6 MB)
Using cached nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl (5
6.5 MB)
Using cached nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl
(124.2 MB)
Using cached nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl
(196.0 MB)
Using cached nvidia_nccl_cu12-2.20.5-py3-none-manylinux2014_x86_64.whl (176.2
MB)
Using cached nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (99 kB)
Downloading Farama_Notifications-0.0.4-py3-none-any.whl (2.5 kB)
Downloading nvidia_nvjitlink_cu12-12.5.82-py3-none-manylinux2014_x86_64.whl
(21.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.3/21.3 MB 76.7 MB/s eta 0:00:0
0
Installing collected packages: farama-notifications, nvidia-nvtx-cu12, nvidia
-nvjitlink-cu12, nvidia-nccl-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvi
dia-cuda-runtime-cu12, nvidia-cuda-nvrtc-cu12, nvidia-cuda-cupti-cu12, nvidia
-cublas-cu12, gymnasium, nvidia-cusparse-cu12, nvidia-cudnn-cu12, nvidia-cuso
lver-cu12, stable_baselines3
Successfully installed farama-notifications-0.0.4 gymnasium-0.29.1 nvidia-cub
las-cu12-12.1.3.1 nvidia-cuda-cupti-cu12-12.1.105 nvidia-cuda-nvrtc-cu12-12.
1.105 nvidia-cuda-runtime-cu12-12.1.105 nvidia-cudnn-cu12-8.9.2.26 nvidia-cuf
ft-cu12-11.0.2.54 nvidia-curand-cu12-10.3.2.106 nvidia-cusolver-cu12-11.4.5.1
07 nvidia-cusparse-cu12-12.1.0.106 nvidia-nccl-cu12-2.20.5 nvidia-nvjitlink-c
u12-12.5.82 nvidia-nvtx-cu12-12.1.105 stable_baselines3-2.3.2
```

In [3]:
```
!pip install gymnasium #Install if necessary or lacking
```

```
Requirement already satisfied: gymnasium in /usr/local/lib/python3.10/dist-pa
ckages (0.29.1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dis
t-packages (from gymnasium) (1.26.4)
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3.1
0/dist-packages (from gymnasium) (2.2.1)
Requirement already satisfied: typing-extensions>=4.3.0 in /usr/local/lib/pyt
hon3.10/dist-packages (from gymnasium) (4.12.2)
Requirement already satisfied: farama-notifications>=0.0.1 in /usr/local/lib/
python3.10/dist-packages (from gymnasium) (0.0.4)
```

IMPORT ALL THE NECESSARY LIBRARIES

In [11]:
```
import gymnasium as gym
import numpy as np
import torch
from stable_baselines3 import DDPG, SAC
from stable_baselines3.common.noise import NormalActionNoise
from stable_baselines3.common.vec_env import DummyVecEnv
from stable_baselines3.common.monitor import Monitor
from stable_baselines3.common.callbacks import BaseCallback
import matplotlib.pyplot as plt

# Define the custom environment for the ISAC system optimization problem
class ISACEnv(gym.Env):
    def __init__(self):
        super(ISACEnv, self).__init__()
        self.action_space = gym.spaces.Box(low=-1, high=1, shape=(2,), dtype=n
        self.observation_space = gym.spaces.Box(low=0, high=1, shape=(4,), dty

    def reset(self, seed=None, options=None):
        super().reset(seed=seed)
        self.state = np.random.rand(4)
        info = {}
        return self.state, info

    def step(self, action):
        reward = -np.sum((action - 0.5)**2)
        self.state = np.random.rand(4)
        terminated = np.random.rand() > 0.95
        truncated = False
        info = {}
        return self.state, reward, terminated, truncated, info

    def render(self, mode='human', close=False):
        pass

# Custom callback to record training progress
class TrainLogger(BaseCallback):
    def __init__(self, verbose=0):
        super(TrainLogger, self).__init__(verbose)
        self.rewards = []

    def _on_step(self) -> bool:
        # Check if ep_info_buffer is not empty
        if self.model.ep_info_buffer:
            last_episode_info = self.model.ep_info_buffer[-1]
            if 'r' in last_episode_info:
                self.rewards.append(last_episode_info['r'])
        return True

# Create and wrap the environment
env = DummyVecEnv([lambda: Monitor(ISACEnv())])

# Create action noise for DDPG
n_actions = env.action_space.shape[-1]
action_noise = NormalActionNoise(mean=np.zeros(n_actions), sigma=0.1 * np.ones

# Train DDPG model
ddpg_logger = TrainLogger()
```

```
ddpg_model = DDPG('MlpPolicy', env, action_noise=action_noise, verbose=1, devi
ddpg_model.learn(total_timesteps=10000, callback=ddpg_logger)

# Train SAC model
sac_logger = TrainLogger()
sac_model = SAC('MlpPolicy', env, verbose=1, device='cpu')
sac_model.learn(total_timesteps=10000, callback=sac_logger)

# Plot the results
plt.figure(figsize=(12, 6))

# Plot DDPG training rewards
plt.subplot(1, 2, 1)
plt.plot(ddpg_logger.rewards, label='DDPG')
plt.xlabel('Timesteps')
plt.ylabel('Rewards')
plt.title('DDPG Training Progress')
plt.legend()

# Plot SAC training rewards
plt.subplot(1, 2, 2)
plt.plot(sac_logger.rewards, label='SAC')
plt.xlabel('Timesteps')
plt.ylabel('Rewards')
plt.title('SAC Training Progress')
plt.legend()

plt.tight_layout()
plt.show()
```

```
Using cpu device
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 13.5     |
|    ep_rew_mean     | -15.1    |
| time/              |          |
|    episodes        | 4        |
|    fps             | 2802     |
|    time_elapsed    | 0        |
|    total_timesteps | 54       |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 14.5     |
|    ep_rew_mean     | -15.6    |
| time/              |          |
|    episodes        | 8        |
|    fps             | 293      |
|    time_elapsed    | 0        |
```

Evaluate and compare the performance

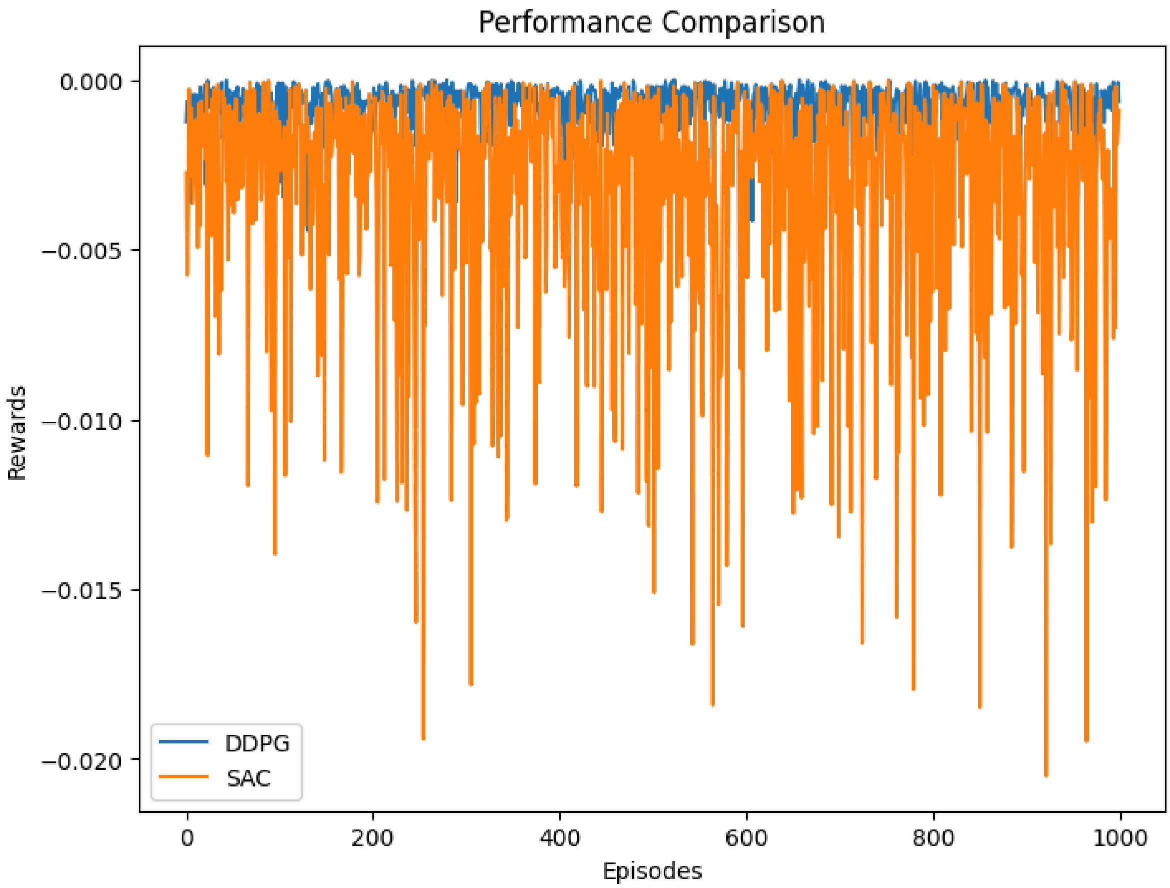
In [12]:
```
# Evaluate and compare performance
ddpg_rewards = []
sac_rewards = []

# Unwrap the environment to access the original env for evaluation
unwrapped_env = env.envs[0]

# Evaluate DDPG model
obs, _ = unwrapped_env.reset()
for _ in range(1000):
    action, _states = ddpg_model.predict(obs, deterministic=True)
    obs, reward, terminated, truncated, info = unwrapped_env.step(action)
    ddpg_rewards.append(reward)
    if terminated:
        obs, _ = unwrapped_env.reset()

# Evaluate SAC model
obs, _ = unwrapped_env.reset()
for _ in range(1000):
    action, _states = sac_model.predict(obs, deterministic=True)
    obs, reward, terminated, truncated, info = unwrapped_env.step(action)
    sac_rewards.append(reward)
    if terminated:
        obs, _ = unwrapped_env.reset()

# Plot the performance comparison
plt.figure(figsize=(8, 6))
plt.plot(ddpg_rewards, label='DDPG')
plt.plot(sac_rewards, label='SAC')
plt.xlabel('Episodes')
plt.ylabel('Rewards')
plt.title('Performance Comparison')
plt.legend()
plt.show()
```



THE END OF OPTIMIZATION USING DRL ALGORITHMS