```
!pip install gym stable_baselines3
```

```
Requirement already satisfied: gym in /usr/local/lib/python3.10/dist-
packages (0.25.2)
Collecting stable_baselines3
  Downloading stable_baselines3-2.2.1-py3-none-any.whl (181 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 181.7/181.7 kB 4.2 MB/s eta
0:00:00
ent already satisfied: numpy>=1.18.0 in
/usr/local/lib/python3.10/dist-packages (from gym) (1.23.5)
Requirement already satisfied: cloudpickle>=1.2.0 in
/usr/local/lib/python3.10/dist-packages (from gym) (2.2.1)
Requirement already satisfied: gym-notices>=0.0.4 in
/usr/local/lib/python3.10/dist-packages (from gym) (0.0.8)
Collecting gymnasium<0.30,>=0.28.1 (from stable_baselines3)
  Downloading gymnasium-0.29.1-py3-none-any.whl (953 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 953.9/953.9 kB 16.2 MB/s eta
0:00:00
ent already satisfied: torch>=1.13 in /usr/local/lib/python3.10/dist-
packages (from stable_baselines3) (2.1.0+cu121)
Requirement already satisfied: pandas in
/usr/local/lib/python3.10/dist-packages (from stable_baselines3)
(1.5.3)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.10/dist-packages (from stable_baselines3)
(3.7.1)
Requirement already satisfied: typing-extensions>=4.3.0 in
/usr/local/lib/python3.10/dist-packages (from gymnasium<0.30,>=0.28.1-
>stable_baselines3) (4.5.0)
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 (2.5 kB)
Requirement already satisfied: filelock in
/usr/local/lib/python3.10/dist-packages (from torch>=1.13-
>stable_baselines3) (3.13.1)
Requirement already satisfied: sympy in
/usr/local/lib/python3.10/dist-packages (from torch>=1.13-
>stable_baselines3) (1.12)
Requirement already satisfied: networkx in
/usr/local/lib/python3.10/dist-packages (from torch>=1.13-
>stable_baselines3) (3.2.1)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.13-
>stable_baselines3) (3.1.2)
Requirement already satisfied: fsspec in
/usr/local/lib/python3.10/dist-packages (from torch>=1.13-
>stable_baselines3) (2023.6.0)
Requirement already satisfied: triton==2.1.0 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.13-
>stable_baselines3) (2.1.0)
```

```
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib-
>stable_baselines3) (1.2.0)
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/python3.10/dist-packages (from matplotlib-
>stable_baselines3) (4.47.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib-
>stable_baselines3) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib-
>stable_baselines3) (23.2)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.10/dist-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) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/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) (2023.3.post1)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7-
>matplotlib->stable_baselines3) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.13-
>stable_baselines3) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in
/usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.13-
>stable_baselines3) (1.3.0)
Installing collected packages: farama-notifications, gymnasium,
stable_baselines3
Successfully installed farama-notifications-0.0.4 gymnasium-0.29.1
stable_baselines3-2.2.1
```

```
!pip install stable-baselines3[extra] gym
```

```
Requirement already satisfied: stable-baselines3[extra] in
/usr/local/lib/python3.10/dist-packages (2.2.1)
Requirement already satisfied: gym in /usr/local/lib/python3.10/dist-
packages (0.25.2)
Requirement already satisfied: gymnasium<0.30,>=0.28.1 in
/usr/local/lib/python3.10/dist-packages (from stable-
baselines3[extra]) (0.29.1)
```

```
Requirement already satisfied: numpy>=1.20 in
/usr/local/lib/python3.10/dist-packages (from stable-
baselines3[extra]) (1.23.5)
Requirement already satisfied: torch>=1.13 in
/usr/local/lib/python3.10/dist-packages (from stable-
baselines3[extra]) (2.1.0+cu121)
Requirement already satisfied: cloudpickle in
/usr/local/lib/python3.10/dist-packages (from stable-
baselines3[extra]) (2.2.1)
Requirement already satisfied: pandas in
/usr/local/lib/python3.10/dist-packages (from stable-
baselines3[extra]) (1.5.3)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.10/dist-packages (from stable-
baselines3[extra]) (3.7.1)
Requirement already satisfied: opencv-python in
/usr/local/lib/python3.10/dist-packages (from stable-
baselines3[extra]) (4.8.0.76)
Requirement already satisfied: pygame in
/usr/local/lib/python3.10/dist-packages (from stable-
baselines3[extra]) (2.5.2)
Requirement already satisfied: tensorboard>=2.9.1 in
/usr/local/lib/python3.10/dist-packages (from stable-
baselines3[extra]) (2.15.1)
Requirement already satisfied: psutil in
/usr/local/lib/python3.10/dist-packages (from stable-
baselines3[extra]) (5.9.5)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-
packages (from stable-baselines3[extra]) (4.66.1)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-
packages (from stable-baselines3[extra]) (13.7.0)
Collecting shimmy[atari]~=1.3.0 (from stable-baselines3[extra])
  Downloading Shimmy-1.3.0-py3-none-any.whl (37 kB)
Requirement already satisfied: pillow in
/usr/local/lib/python3.10/dist-packages (from stable-
baselines3[extra]) (9.4.0)
Collecting autorom[accept-rom-license]~=0.6.1 (from stable-
baselines3[extra])
  Downloading AutoROM-0.6.1-py3-none-any.whl (9.4 kB)
Requirement already satisfied: gym-notices>=0.0.4 in
/usr/local/lib/python3.10/dist-packages (from gym) (0.0.8)
Requirement already satisfied: click in
/usr/local/lib/python3.10/dist-packages (from autorom[accept-rom-
license]~=0.6.1->stable-baselines3[extra]) (8.1.7)
Requirement already satisfied: requests in
/usr/local/lib/python3.10/dist-packages (from autorom[accept-rom-
license]~=0.6.1->stable-baselines3[extra]) (2.31.0)
Collecting AutoROM.accept-rom-license (from autorom[accept-rom-
license]~=0.6.1->stable-baselines3[extra])
```

```
  Downloading AutoROM.accept-rom-license-0.6.1.tar.gz (434 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 434.7/434.7 kB 6.7 MB/s eta
0:00:00
ents to build wheel ... etadata (pyproject.toml) ... ent already
satisfied: typing-extensions>=4.3.0 in /usr/local/lib/python3.10/dist-
packages (from gymnasium<0.30,>=0.28.1->stable-baselines3[extra])
(4.5.0)
Requirement already satisfied: farama-notifications>=0.0.1 in
/usr/local/lib/python3.10/dist-packages (from gymnasium<0.30,>=0.28.1-
>stable-baselines3[extra]) (0.0.4)
Collecting ale-py~=0.8.1 (from shimmy[atari]~=1.3.0->stable-
baselines3[extra])
  Downloading ale_py-0.8.1-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.7 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 13.8 MB/s eta
0:00:00
ent already satisfied: absl-py>=0.4 in /usr/local/lib/python3.10/dist-
packages (from tensorboard>=2.9.1->stable-baselines3[extra]) (1.4.0)
Requirement already satisfied: grpcio>=1.48.2 in
/usr/local/lib/python3.10/dist-packages (from tensorboard>=2.9.1-
>stable-baselines3[extra]) (1.60.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.10/dist-packages (from tensorboard>=2.9.1-
>stable-baselines3[extra]) (2.17.3)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in
/usr/local/lib/python3.10/dist-packages (from tensorboard>=2.9.1-
>stable-baselines3[extra]) (1.2.0)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from tensorboard>=2.9.1-
>stable-baselines3[extra]) (3.5.1)
Requirement already satisfied: protobuf<4.24,>=3.19.6 in
/usr/local/lib/python3.10/dist-packages (from tensorboard>=2.9.1-
>stable-baselines3[extra]) (3.20.3)
Requirement already satisfied: setuptools>=41.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorboard>=2.9.1-
>stable-baselines3[extra]) (67.7.2)
Requirement already satisfied: six>1.9 in
/usr/local/lib/python3.10/dist-packages (from tensorboard>=2.9.1-
>stable-baselines3[extra]) (1.16.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0
in /usr/local/lib/python3.10/dist-packages (from tensorboard>=2.9.1-
>stable-baselines3[extra]) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from tensorboard>=2.9.1-
>stable-baselines3[extra]) (3.0.1)
Requirement already satisfied: filelock in
/usr/local/lib/python3.10/dist-packages (from torch>=1.13->stable-
baselines3[extra]) (3.13.1)
Requirement already satisfied: sympy in
```

```
/usr/local/lib/python3.10/dist-packages (from torch>=1.13->stable-
baselines3[extra]) (1.12)
Requirement already satisfied: networkx in
/usr/local/lib/python3.10/dist-packages (from torch>=1.13->stable-
baselines3[extra]) (3.2.1)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.13->stable-
baselines3[extra]) (3.1.2)
Requirement already satisfied: fsspec in
/usr/local/lib/python3.10/dist-packages (from torch>=1.13->stable-
baselines3[extra]) (2023.6.0)
Requirement already satisfied: triton==2.1.0 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.13->stable-
baselines3[extra]) (2.1.0)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->stable-
baselines3[extra]) (1.2.0)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->stable-
baselines3[extra]) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->stable-
baselines3[extra]) (4.47.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->stable-
baselines3[extra]) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->stable-
baselines3[extra]) (23.2)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->stable-
baselines3[extra]) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->stable-
baselines3[extra]) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas->stable-
baselines3[extra]) (2023.3.post1)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich->stable-
baselines3[extra]) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich->stable-
baselines3[extra]) (2.16.1)
Requirement already satisfied: importlib-resources in
/usr/local/lib/python3.10/dist-packages (from ale-py~=0.8.1-
>shimmy[atari]~=1.3.0->stable-baselines3[extra]) (6.1.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
```

```
>tensorboard>=2.9.1->stable-baselines3[extra]) (5.3.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard>=2.9.1->stable-baselines3[extra]) (0.3.0)
Requirement already satisfied: rsa<5,>=3.1.4 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard>=2.9.1->stable-baselines3[extra]) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth-
oauthlib<2,>=0.5->tensorboard>=2.9.1->stable-baselines3[extra])
(1.3.1)
Requirement already satisfied: mdurl~=0.1 in
/usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0-
>rich->stable-baselines3[extra]) (0.1.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests-
>autorom[accept-rom-license]~=0.6.1->stable-baselines3[extra]) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests-
>autorom[accept-rom-license]~=0.6.1->stable-baselines3[extra]) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests-
>autorom[accept-rom-license]~=0.6.1->stable-baselines3[extra]) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests-
>autorom[accept-rom-license]~=0.6.1->stable-baselines3[extra])
(2023.11.17)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1-
>tensorboard>=2.9.1->stable-baselines3[extra]) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in
/usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.13-
>stable-baselines3[extra]) (1.3.0)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in
/usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1-
>google-auth<3,>=1.6.3->tensorboard>=2.9.1->stable-baselines3[extra])
(0.5.1)
Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from requests-
oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5->tensorboard>=2.9.1-
>stable-baselines3[extra]) (3.2.2)
Building wheels for collected packages: AutoROM.accept-rom-license
  Building wheel for AutoROM.accept-rom-license (pyproject.toml) ... -
license: filename=AutoROM.accept_rom_license-0.6.1-py3-none-any.whl
size=446660
sha256=61ea886a6ab05a3e22ace0ce2eddecccbc7f6d8d55a662cac0215444018f195
3
  Stored in directory:
/root/.cache/pip/wheels/6b/1b/ef/a43ff1a2f1736d5711faa1ba4c1f61be1131b
```

```
8899e6a057811
Successfully built AutoROM.accept-rom-license
Installing collected packages: ale-py, shimmy, AutoROM.accept-rom-
license, autorom
Successfully installed AutoROM.accept-rom-license-0.6.1 ale-py-0.8.1
autorom-0.6.1 shimmy-1.3.0
```

```
!pip install 'shimmy>=0.2.1'
```

```
Requirement already satisfied: shimmy>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (1.3.0)
Requirement already satisfied: numpy>=1.18.0 in
/usr/local/lib/python3.10/dist-packages (from shimmy>=0.2.1) (1.23.5)
Requirement already satisfied: gymnasium>=0.27.0 in
/usr/local/lib/python3.10/dist-packages (from shimmy>=0.2.1) (0.29.1)
Requirement already satisfied: cloudpickle>=1.2.0 in
/usr/local/lib/python3.10/dist-packages (from gymnasium>=0.27.0-
>shimmy>=0.2.1) (2.2.1)
Requirement already satisfied: typing-extensions>=4.3.0 in
/usr/local/lib/python3.10/dist-packages (from gymnasium>=0.27.0-
>shimmy>=0.2.1) (4.5.0)
Requirement already satisfied: farama-notifications>=0.0.1 in
/usr/local/lib/python3.10/dist-packages (from gymnasium>=0.27.0-
>shimmy>=0.2.1) (0.0.4)
```

```
import gym
from gym import spaces
import numpy as np
from stable_baselines3 import PPO

class JobSchedulingEnv(gym.Env):
    def __init__(self):
        super(JobSchedulingEnv, self).__init__()
        self.num_jobs = 6
        self.job_durations = [2, 3, 5, 6, 2, 3]
        self.action_space = spaces.Discrete(self.num_jobs)
        self.observation_space = spaces.MultiBinary(self.num_jobs)
        self.max_steps = 100  # Maximum number of steps per episode
        self.reset()

    def step(self, action):
        # Toggle the machine assignment for the selected job
        self.state[action] = 1 - self.state[action]

        # Calculate the makespan for each machine
        makespan_m1 = sum([duration for i, duration in
enumerate(self.job_durations) if self.state[i] == 0])
        makespan_m2 = sum([duration for i, duration in
enumerate(self.job_durations) if self.state[i] == 1])
        new_makespan = max(makespan_m1, makespan_m2)
```

```
        # Calculate reward based on the change in makespan
        reward = self.current_makespan - new_makespan
        self.current_makespan = new_makespan

        # Increment the step count and check for termination
        self.current_step += 1
        done = self.current_step >= self.max_steps

        return self.state, reward, done, {}

    def reset(self):
        self.state = np.zeros(self.num_jobs, dtype=int)
        self.current_makespan = sum(self.job_durations)
        self.current_step = 0
        return self.state

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

        # Assign jobs to each machine based on the current state
        jobs_on_m1 = [f"J{i+1}" for i in range(self.num_jobs) if
self.state[i] == 0]
        jobs_on_m2 = [f"J{i+1}" for i in range(self.num_jobs) if
self.state[i] == 1]

        # Calculate makespan for each machine
        makespan_m1 = sum([duration for i, duration in
enumerate(self.job_durations) if self.state[i] == 0])
        makespan_m2 = sum([duration for i, duration in
enumerate(self.job_durations) if self.state[i] == 1])

        # Print the scheduling status
        print(f"Machine 1 (M1) - Jobs: {', '.join(jobs_on_m1)} |
Makespan: {makespan_m1} minutes")
        print(f"Machine 2 (M2) - Jobs: {', '.join(jobs_on_m2)} |
Makespan: {makespan_m2} minutes")
        print("-" * 50)

    # Method to set a specific initial state (optional)
    def set_initial_state(self, initial_state):
        if len(initial_state) == self.num_jobs:
            self.state = np.array(initial_state, dtype=int)

# Initialize the environment and the model
env = JobSchedulingEnv()
model = PPO("MlpPolicy", env, verbose=1)

# Train the model
```

```
model.learn(total_timesteps=10000)

# Optionally, set a specific problem before testing
# env.set_initial_state([0, 0, 1, 1, 0, 1]) # Example initial state

# Test the trained agent
obs = env.reset()
for i in range(1000):
    action, _states = model.predict(obs, deterministic=True)
    obs, rewards, dones, info = env.step(action)
    if dones:
        obs = env.reset()
    env.render()

# Save the model
model.save("job_scheduling_model")
```

```
from stable_baselines3 import PPO

# Load the trained model
model = PPO.load("job_scheduling_model")

# Update the job durations for the new problem
new_job_durations = [2.30, 4.12, 7, 6, 2, 3]  # Replace with your job
durations

# Create a new environment with the updated job durations
class NewJobSchedulingEnv(JobSchedulingEnv):
    def __init__(self):
        super().__init__()
        self.job_durations = new_job_durations  # Update the job
durations

# Initialize the new environment
new_env = NewJobSchedulingEnv()

# Initialize variables to track the optimal solution
optimal_makespan = float('inf')
optimal_state = None

# Test the trained agent on the new environment
obs = new_env.reset()
for i in range(1000):
    action, _states = model.predict(obs, deterministic=True)
    obs, rewards, dones, info = new_env.step(action)

    # Check if the current solution is better than the best found so
far
    if new_env.current_makespan < optimal_makespan:
```

```
        optimal_makespan = new_env.current_makespan
        optimal_state = obs.copy()

    if dones:
        obs = new_env.reset()

# Print the optimal solution
jobs_on_m1 = [f"J{i+1}" for i in range(new_env.num_jobs) if
optimal_state[i] == 0]
jobs_on_m2 = [f"J{i+1}" for i in range(new_env.num_jobs) if
optimal_state[i] == 1]

print("Optimal Solution:")
print(f"Machine 1 (M1) - Jobs: {', '.join(jobs_on_m1)} | Makespan:
{sum(new_env.job_durations[i] for i in range(new_env.num_jobs) if
optimal_state[i] == 0)} minutes")
print(f"Machine 2 (M2) - Jobs: {', '.join(jobs_on_m2)} | Makespan:
{sum(new_env.job_durations[i] for i in range(new_env.num_jobs) if
optimal_state[i] == 1)} minutes")

import gym
from gym import spaces
import numpy as np
import random
from stable_baselines3.common.env_util import make_vec_env
from stable_baselines3 import PPO

class JobSchedulingEnv(gym.Env):
    def __init__(self, num_jobs=6, job_durations=[2, 3, 5, 6, 2, 3],
num_machines=2):
        super(JobSchedulingEnv, self).__init__()
        self.num_jobs = num_jobs
        self.job_durations = job_durations
        self.num_machines = num_machines

        # Action space: Each element in the action array represents a
job's assigned machine
        self.action_space = spaces.MultiDiscrete([num_machines] *
num_jobs)

        # Observation space: Each job's current machine assignment
        self.observation_space = spaces.MultiDiscrete([num_machines] *
num_jobs)

        self.max_steps = 2000
        self.reset()

    def step(self, action):
        # Save the previous maximum makespan before updating the state
        prev_max_makespan = max([sum(self.job_durations[j] for j in
```

```
range(self.num_jobs) if self.state[j] == m) for m in
range(self.num_machines)])

        # Update the state based on the action
        self.state = action

        # Calculate the new makespan for each machine
        makespans = [sum(self.job_durations[j] for j in
range(self.num_jobs) if self.state[j] == m) for m in
range(self.num_machines)]
        new_max_makespan = max(makespans)

        # Calculate reward based on the change in the maximum makespan
        reward = prev_max_makespan - new_max_makespan

        # Increment the step count and check if the episode is done
        self.current_step += 1
        done = self.current_step >= self.max_steps

        return self.state, reward, done, {}


    def reset(self):
        self.state = np.zeros(self.num_jobs, dtype=int)
        self.current_makespan = sum(self.job_durations)
        self.current_step = 0
        return self.state

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

        for m in range(self.num_machines):
            jobs_on_machine = [f"J{i+1}" for i in range(self.num_jobs)
if self.state[i] == m]
            makespan = sum(self.job_durations[i] for i in
range(self.num_jobs) if self.state[i] == m)
            print(f"Machine {m+1} - Jobs: {', '.join(jobs_on_machine)}
| Makespan: {makespan} minutes")
        print("-" * 50)



number_of_epochs = 10  # Define the number of epochs
timesteps_per_epoch = 2000  # Define the number of timesteps per epoch
num_jobs = 6
num_machines = 3

# Initialize the environment with initial job durations
initial_job_durations = [random.uniform(1, 12) for _ in
```

```
range(num_jobs)]
env = JobSchedulingEnv(num_jobs=num_jobs,
job_durations=initial_job_durations, num_machines=num_machines)
env = make_vec_env(lambda: env, n_envs=1)

# Initialize the model
model = PPO("MlpPolicy", env, learning_rate=0.00025, n_steps=2048,
batch_size=64,
            gamma=0.99, gae_lambda=0.95, clip_range=0.2,
ent_coef=0.01,
            verbose=1,
tensorboard_log="./ppo_job_scheduling_tensorboard/")

# Train the model over multiple epochs with different job durations
for epoch in range(number_of_epochs):
    # Generate new job durations for this epoch
    new_job_durations = [random.uniform(1, 12) for _ in
range(num_jobs)]

    # Update the environment with new job durations
    env.envs[0].env.job_durations = new_job_durations

    # Continue training the model
    model.learn(total_timesteps=timesteps_per_epoch)

    # Optional: Save the model after each epoch
model_filename =
f"job_scheduling_model_epoch_{num_machines}machines_{num_jobs}jobs"
model.save(model_filename)

import gym
from gym import spaces
import numpy as np
import random
from stable_baselines3.common.env_util import make_vec_env
from stable_baselines3 import PPO

from stable_baselines3.common.callbacks import EvalCallback,
CheckpointCallback


class JobSchedulingEnv(gym.Env):
    def __init__(self, num_jobs=6, job_durations=[2, 3, 5, 6, 2, 3],
num_machines=2):
        super(JobSchedulingEnv, self).__init__()
        self.num_jobs = num_jobs
        self.job_durations = job_durations
        self.num_machines = num_machines
        self.action_space = spaces.MultiDiscrete([num_machines] *
num_jobs)
```

```
        self.observation_space = spaces.MultiDiscrete([num_machines] *
num_jobs)
        self.max_steps = 2000
        self.reset()

    def step(self, action):
        prev_max_makespan = max([sum(self.job_durations[j] for j in
range(self.num_jobs) if self.state[j] == m) for m in
range(self.num_machines)])
        self.state = action
        makespans = [sum(self.job_durations[j] for j in
range(self.num_jobs) if self.state[j] == m) for m in
range(self.num_machines)]
        new_max_makespan = max(makespans)
        reward = prev_max_makespan - new_max_makespan
        self.current_step += 1
        done = self.current_step >= self.max_steps
        return self.state, reward, done, {}

    def reset(self):
        self.state = np.zeros(self.num_jobs, dtype=int)
        self.current_makespan = sum(self.job_durations)
        self.current_step = 0
        return self.state

    def render(self, mode='human', close=False):
        if close:
            return
        for m in range(self.num_machines):
            jobs_on_machine = [f"J{i+1}" for i in range(self.num_jobs)
if self.state[i] == m]
            makespan = sum(self.job_durations[i] for i in
range(self.num_jobs) if self.state[i] == m)
            print(f"Machine {m+1} - Jobs: {', '.join(jobs_on_machine)}
| Makespan: {makespan} minutes")
        print("-" * 50)

number_of_epochs = 50
timesteps_per_epoch = 2000
num_jobs = 5
num_machines = 3



training_scenarios = [
    [random.uniform(1, 12) for _ in range(num_jobs)] for _ in
range(number_of_epochs)
]

initial_job_durations = training_scenarios[0]
```

```python
env = JobSchedulingEnv(num_jobs=num_jobs,
job_durations=initial_job_durations, num_machines=num_machines)
env = make_vec_env(lambda: env, n_envs=1)

model = PPO("MlpPolicy", env, learning_rate=0.00025, n_steps=2048,
batch_size=64,
            gamma=0.99, gae_lambda=0.95, clip_range=0.2,
ent_coef=0.01,
            verbose=1,
tensorboard_log="./ppo_job_scheduling_tensorboard/")
# Evaluation callback for logging performance and progress
eval_env = make_vec_env(lambda: JobSchedulingEnv(num_jobs=num_jobs,
job_durations=initial_job_durations, num_machines=num_machines),
n_envs=1)
eval_callback = EvalCallback(eval_env, best_model_save_path='./logs/',
                             log_path='./logs/', eval_freq=500,
                             deterministic=True, render=False)

# Checkpoint callback for saving the model
checkpoint_callback = CheckpointCallback(save_freq=1000,
save_path='./logs/',
                                         name_prefix='rl_model')


for epoch, new_job_durations in enumerate(training_scenarios):
    env.envs[0].env.job_durations = new_job_durations
    model.learn(total_timesteps=timesteps_per_epoch)

model_filename =
f"job_scheduling_model_epoch_{num_machines}machines_{num_jobs}jobs"
model.save(model_filename)
```

```
Using cuda device
Logging to ./ppo_job_scheduling_tensorboard/PPO_2
```

```
/usr/local/lib/python3.10/dist-packages/stable_baselines3/common/
vec_env/patch_gym.py:49: UserWarning: You provided an OpenAI Gym
environment. We strongly recommend transitioning to Gymnasium
environments. Stable-Baselines3 is automatically wrapping your
environments in a compatibility layer, which could potentially cause
issues.
  warnings.warn(
```

```
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 20.6     |
| time/              |          |
|    fps             | 230      |
|    iterations      | 1        |
```

```
|    time_elapsed    | 8        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_3
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 20.6     |
| time/              |          |
|    fps             | 230      |
|    iterations      | 1        |
|    time_elapsed    | 8        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_4
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 21.5     |
| time/              |          |
|    fps             | 228      |
|    iterations      | 1        |
|    time_elapsed    | 8        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_5
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 8.89     |
| time/              |          |
|    fps             | 226      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_6
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 21.5     |
| time/              |          |
|    fps             | 227      |
|    iterations      | 1        |
|    time_elapsed    | 8        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_7
---------------------------------
```

```
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 21.4     |
| time/              |          |
|    fps             | 228      |
|    iterations      | 1        |
|    time_elapsed    | 8        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_8
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 14.2     |
| time/              |          |
|    fps             | 229      |
|    iterations      | 1        |
|    time_elapsed    | 8        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_9
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 13.3     |
| time/              |          |
|    fps             | 224      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_10
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 21.4     |
| time/              |          |
|    fps             | 225      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_11
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 14       |
| time/              |          |
|    fps             | 224      |
```

```
|    iterations      | 1          |
|    time_elapsed    | 9          |
|    total_timesteps | 2048       |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_12
---------------------------------
| rollout/           |            |
|    ep_len_mean     | 2e+03      |
|    ep_rew_mean     | 14.8       |
| time/              |            |
|    fps             | 226        |
|    iterations      | 1          |
|    time_elapsed    | 9          |
|    total_timesteps | 2048       |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_13
---------------------------------
| rollout/           |            |
|    ep_len_mean     | 2e+03      |
|    ep_rew_mean     | 21.4       |
| time/              |            |
|    fps             | 228        |
|    iterations      | 1          |
|    time_elapsed    | 8          |
|    total_timesteps | 2048       |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_14
---------------------------------
| rollout/           |            |
|    ep_len_mean     | 2e+03      |
|    ep_rew_mean     | 19.8       |
| time/              |            |
|    fps             | 227        |
|    iterations      | 1          |
|    time_elapsed    | 8          |
|    total_timesteps | 2048       |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_15
---------------------------------
| rollout/           |            |
|    ep_len_mean     | 2e+03      |
|    ep_rew_mean     | 14.7       |
| time/              |            |
|    fps             | 230        |
|    iterations      | 1          |
|    time_elapsed    | 8          |
|    total_timesteps | 2048       |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_16
```

```
----------------------------------
| rollout/            |          |
|    ep_len_mean      | 2e+03    |
|    ep_rew_mean      | 20       |
| time/               |          |
|    fps              | 236      |
|    iterations       | 1        |
|    time_elapsed     | 8        |
|    total_timesteps  | 2048     |
----------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_17
----------------------------------
| rollout/            |          |
|    ep_len_mean      | 2e+03    |
|    ep_rew_mean      | 14       |
| time/               |          |
|    fps              | 238      |
|    iterations       | 1        |
|    time_elapsed     | 8        |
|    total_timesteps  | 2048     |
----------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_18
----------------------------------
| rollout/            |          |
|    ep_len_mean      | 2e+03    |
|    ep_rew_mean      | 19.8     |
| time/               |          |
|    fps              | 237      |
|    iterations       | 1        |
|    time_elapsed     | 8        |
|    total_timesteps  | 2048     |
----------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_19
----------------------------------
| rollout/            |          |
|    ep_len_mean      | 2e+03    |
|    ep_rew_mean      | 5.81     |
| time/               |          |
|    fps              | 239      |
|    iterations       | 1        |
|    time_elapsed     | 8        |
|    total_timesteps  | 2048     |
----------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_20
----------------------------------
| rollout/            |          |
|    ep_len_mean      | 2e+03    |
|    ep_rew_mean      | 19.8     |
| time/               |          |
```

```
|    fps             | 239      |
|    iterations      | 1        |
|    time_elapsed    | 8        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_21
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 19.8     |
| time/              |          |
|    fps             | 238      |
|    iterations      | 1        |
|    time_elapsed    | 8        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_22
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 16.2     |
| time/              |          |
|    fps             | 240      |
|    iterations      | 1        |
|    time_elapsed    | 8        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_23
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 22.2     |
| time/              |          |
|    fps             | 237      |
|    iterations      | 1        |
|    time_elapsed    | 8        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_24
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 20.6     |
| time/              |          |
|    fps             | 232      |
|    iterations      | 1        |
|    time_elapsed    | 8        |
|    total_timesteps | 2048     |
---------------------------------
```

```
Logging to ./ppo_job_scheduling_tensorboard/PPO_25
----------------------------------
| rollout/           |           |
|    ep_len_mean     | 2e+03     |
|    ep_rew_mean     | 22.2      |
| time/              |           |
|    fps             | 229       |
|    iterations      | 1         |
|    time_elapsed    | 8         |
|    total_timesteps | 2048      |
----------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_26
----------------------------------
| rollout/           |           |
|    ep_len_mean     | 2e+03     |
|    ep_rew_mean     | 14        |
| time/              |           |
|    fps             | 227       |
|    iterations      | 1         |
|    time_elapsed    | 9         |
|    total_timesteps | 2048      |
----------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_27
----------------------------------
| rollout/           |           |
|    ep_len_mean     | 2e+03     |
|    ep_rew_mean     | 7.47      |
| time/              |           |
|    fps             | 225       |
|    iterations      | 1         |
|    time_elapsed    | 9         |
|    total_timesteps | 2048      |
----------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_28
----------------------------------
| rollout/           |           |
|    ep_len_mean     | 2e+03     |
|    ep_rew_mean     | 15.6      |
| time/              |           |
|    fps             | 223       |
|    iterations      | 1         |
|    time_elapsed    | 9         |
|    total_timesteps | 2048      |
----------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_29
----------------------------------
| rollout/           |           |
|    ep_len_mean     | 2e+03     |
|    ep_rew_mean     | 14        |
```

```
| time/              |          |
|    fps             | 226      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_30
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 20       |
| time/              |          |
|    fps             | 224      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_31
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 15.6     |
| time/              |          |
|    fps             | 224      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_32
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 13.3     |
| time/              |          |
|    fps             | 224      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_33
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 7.47     |
| time/              |          |
|    fps             | 226      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
|    total_timesteps | 2048     |
```

```
-----------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_34
-----------------------------------
| rollout/           |            |
|    ep_len_mean     | 2e+03      |
|    ep_rew_mean     | 20         |
| time/              |            |
|    fps             | 227        |
|    iterations      | 1          |
|    time_elapsed    | 9          |
|    total_timesteps | 2048       |
-----------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_35
-----------------------------------
| rollout/           |            |
|    ep_len_mean     | 2e+03      |
|    ep_rew_mean     | 20         |
| time/              |            |
|    fps             | 225        |
|    iterations      | 1          |
|    time_elapsed    | 9          |
|    total_timesteps | 2048       |
-----------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_36
-----------------------------------
| rollout/           |            |
|    ep_len_mean     | 2e+03      |
|    ep_rew_mean     | 14.7       |
| time/              |            |
|    fps             | 224        |
|    iterations      | 1          |
|    time_elapsed    | 9          |
|    total_timesteps | 2048       |
-----------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_37
-----------------------------------
| rollout/           |            |
|    ep_len_mean     | 2e+03      |
|    ep_rew_mean     | 14.7       |
| time/              |            |
|    fps             | 225        |
|    iterations      | 1          |
|    time_elapsed    | 9          |
|    total_timesteps | 2048       |
-----------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_38
-----------------------------------
| rollout/           |            |
|    ep_len_mean     | 2e+03      |
```

```
|    ep_rew_mean     | 13.3     |
| time/              |          |
|    fps             | 224      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_39
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 13.1     |
| time/              |          |
|    fps             | 225      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_40
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 16.4     |
| time/              |          |
|    fps             | 225      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_41
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 16.4     |
| time/              |          |
|    fps             | 223      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_42
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 14.7     |
| time/              |          |
|    fps             | 223      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
```

```
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_43
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 13.1     |
| time/              |          |
|    fps             | 227      |
|    iterations      | 1        |
|    time_elapsed    | 8        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_44
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 21.5     |
| time/              |          |
|    fps             | 226      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_45
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 15.6     |
| time/              |          |
|    fps             | 227      |
|    iterations      | 1        |
|    time_elapsed    | 8        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_46
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 22.2     |
| time/              |          |
|    fps             | 226      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_47
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
```

```
|    ep_rew_mean     | 21.4     |
| time/              |          |
|    fps             | 224      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_48
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 8.89     |
| time/              |          |
|    fps             | 226      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_49
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 20       |
| time/              |          |
|    fps             | 225      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_50
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 22.2     |
| time/              |          |
|    fps             | 226      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
|    total_timesteps | 2048     |
---------------------------------
Logging to ./ppo_job_scheduling_tensorboard/PPO_51
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 2e+03    |
|    ep_rew_mean     | 15.6     |
| time/              |          |
|    fps             | 224      |
|    iterations      | 1        |
|    time_elapsed    | 9        |
```

```
|    total_timesteps | 2048     |
---------------------------------
```

```
# Load the model- working testing model
loaded_model = PPO.load("job_scheduling_model_epoch_3machines_5jobs")

# Create an instance of the environment for testing
num_jobs = 5
job_durations = [10, 3, 20, 8, 1]  # These should match the training
setup
num_machines = 3
test_env = JobSchedulingEnv(num_jobs=num_jobs,
job_durations=job_durations, num_machines=num_machines)

# Initialize variables to track the optimal solution
optimal_makespan = float('inf')
optimal_state = None

# Run the model to find the optimal solution
obs = test_env.reset()
for _ in range(20000):
    # Introduce a small probability of random action to allow
exploration
    if random.random() < 0.05:  # 5% chance of random action
        action = test_env.action_space.sample()
    else:
        action, _states = loaded_model.predict(obs,
deterministic=False)

    obs, _, dones, _ = test_env.step(action)

    # Track the best solution
    if test_env.current_makespan < optimal_makespan:
        optimal_makespan = test_env.current_makespan
        optimal_state = obs.copy()

    if dones:
        obs = test_env.reset()

# Print the optimal solution
print("Optimal Schedule:")
for m in range(num_machines):
    jobs_on_machine = [f"J{i+1}" for i in range(num_jobs) if
optimal_state[i] == m]
    makespan = sum(test_env.job_durations[i] for i in range(num_jobs)
if optimal_state[i] == m)
    print(f"Machine {m+1} - Jobs: {', '.join(jobs_on_machine)} |
Makespan: {makespan} minutes")
```

```
Optimal Schedule:
Machine 1 - Jobs: J2, J4, J5 | Makespan: 12 minutes
Machine 2 - Jobs: J1 | Makespan: 10 minutes
Machine 3 - Jobs: J3 | Makespan: 20 minutes
```

```
%load_ext tensorboard
%tensorboard --logdir ./ppo_job_scheduling_tensorboard/

# Load the model itioal development
loaded_model = PPO.load("job_scheduling_model", env=env)

# Create an instance of the original environment for rendering
render_env = JobSchedulingEnv(num_jobs=8, job_durations=[2, 1, 4, 3,
5, 2, 6, 3], num_machines=3)

# Test the loaded model
obs = env.reset()
for i in range(200000):
    action, _states = loaded_model.predict(obs, deterministic=True)
    obs, rewards, dones, info = env.step(action)

    # Synchronize the state of the rendering environment
    render_env.state = env.get_attr("state")[0]
    render_env.current_makespan = env.get_attr("current_makespan")[0]

    # Use the render method of the original environment
    render_env.render()

    if dones:
        obs = env.reset()

import gym
from gym import spaces
import numpy as np
import random
from stable_baselines3 import PPO
from stable_baselines3.common.env_util import make_vec_env

class JobSchedulingEnv(gym.Env):
    def __init__(self, num_jobs=6, job_durations=[2, 3, 5, 6, 2, 3],
num_machines=2):
        super(JobSchedulingEnv, self).__init__()
        self.num_jobs = num_jobs
        self.job_durations = job_durations
        self.num_machines = num_machines
        self.action_space = spaces.MultiDiscrete([num_machines] *
num_jobs)
        self.observation_space = spaces.MultiDiscrete([num_machines] *
num_jobs)
        self.max_steps = 2000
        self.reset()
```

```
    def step(self, action):
        prev_max_makespan = max([sum(self.job_durations[j] for j in
range(self.num_jobs) if self.state[j] == m) for m in
range(self.num_machines)])
        self.state = action
        makespans = [sum(self.job_durations[j] for j in
range(self.num_jobs) if self.state[j] == m) for m in
range(self.num_machines)]
        new_max_makespan = max(makespans)
        reward = prev_max_makespan - new_max_makespan
        self.current_step += 1
        done = self.current_step >= self.max_steps
        return self.state, reward, done, {}

    def reset(self):
        self.state = np.zeros(self.num_jobs, dtype=int)
        self.current_makespan = sum(self.job_durations)
        self.current_step = 0
        return self.state

    def render(self, mode='human', close=False):
        if close:
            return
        for m in range(self.num_machines):
            jobs_on_machine = [f"J{i+1}" for i in range(self.num_jobs)
if self.state[i] == m]
            makespan = sum(self.job_durations[i] for i in
range(self.num_jobs) if self.state[i] == m)
            print(f"Machine {m+1} - Jobs: {', '.join(jobs_on_machine)}
| Makespan: {makespan} minutes")
        print("-" * 50)

# Initialize your environment parameters
number_of_epochs = 100
timesteps_per_epoch = 2000
num_jobs = 5
num_machines = 3

# Epsilon-Greedy Parameters
epsilon_start = 1.0
epsilon_end = 0.01
epsilon_decay = 0.995

def select_action(model, observation, epsilon, env):
    if random.random() < epsilon:
        # Generate a random action for each environment in the batch
        return [env.action_space.sample() for _ in
range(env.num_envs)]
    else:
```

```
        # Predict action using the model for each environment in the
batch
        return model.predict(observation, deterministic=True)[0]


# Generating training scenarios
training_scenarios = [
    [random.uniform(1, 12) for _ in range(num_jobs)] for _ in
range(number_of_epochs)
]

initial_job_durations = training_scenarios[0]
env = JobSchedulingEnv(num_jobs=num_jobs,
job_durations=initial_job_durations, num_machines=num_machines)
env = make_vec_env(lambda: env, n_envs=1)

# Initialize the PPO model
model = PPO("MlpPolicy", env, learning_rate=0.0025, n_steps=2048,
batch_size=64,
            gamma=0.99, gae_lambda=0.95, clip_range=0.2,
ent_coef=0.01,
            verbose=1,
tensorboard_log="./ppo_job_scheduling_tensorboard/")

# Training loop with epsilon-greedy exploration
epsilon = epsilon_start
for epoch in range(number_of_epochs):
    obs = env.reset()
    for step in range(timesteps_per_epoch):
        action = select_action(model, obs, epsilon, env)
        obs, rewards, dones, infos = env.step(action)
        # ... (additional code for your training step) ...

    # Decay epsilon
    epsilon = max(epsilon_end, epsilon_decay * epsilon)

# Saving the model
model_filename =
f"job_scheduling_model_epoch_{num_machines}machines_{num_jobs}jobs"
model.save(model_filename)

# ... [Your existing code for loading the model and testing] ...
```

```
Using cuda device
```

```
# Load the model- working testing model
loaded_model = PPO.load("job_scheduling_model_epoch_3machines_5jobs")

# Create an instance of the environment for testing
num_jobs = 5
```

```
job_durations = [10, 3, 20, 8, 1]  # These should match the training
setup
num_machines = 3
test_env = JobSchedulingEnv(num_jobs=num_jobs,
job_durations=job_durations, num_machines=num_machines)

# Initialize variables to track the optimal solution
optimal_makespan = float('inf')
optimal_state = None

# Run the model to find the optimal solution
obs = test_env.reset()
for _ in range(1000):
    # Introduce a small probability of random action to allow
exploration
    if random.random() < 0.05:  # 5% chance of random action
        action = test_env.action_space.sample()
    else:
        action, _states = loaded_model.predict(obs,
deterministic=False)

    obs, _, dones, _ = test_env.step(action)

    # Track the best solution
    if test_env.current_makespan < optimal_makespan:
        optimal_makespan = test_env.current_makespan
        optimal_state = obs.copy()

    if dones:
        obs = test_env.reset()

# Print the optimal solution
print("Optimal Schedule:")
for m in range(num_machines):
    jobs_on_machine = [f"J{i+1}" for i in range(num_jobs) if
optimal_state[i] == m]
    makespan = sum(test_env.job_durations[i] for i in range(num_jobs)
if optimal_state[i] == m)
    print(f"Machine {m+1} - Jobs: {', '.join(jobs_on_machine)} |
Makespan: {makespan} minutes")

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
last)
<ipython-input-1-e5d3635149aa> in <cell line: 2>()
      1 # Load the model- working testing model
----> 2 loaded_model =
PPO.load("job_scheduling_model_epoch_3machines_5jobs")
      3
```

```
      4 # Create an instance of the environment for testing
      5 num_jobs = 5

NameError: name 'PPO' is not defined
```

```
!pip install pulp
```

```
Collecting pulp
  Downloading PuLP-2.7.0-py3-none-any.whl (14.3 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.3/14.3 MB 38.8 MB/s eta
0:00:00
```

```
import random

# Parameters
num_jobs = 5
job_durations = [10, 3, 20, 8, 1]
num_machines = 3
population_size = 50
generations = 100
crossover_rate = 0.8
mutation_rate = 0.1

# Initialize population
def initialize_population(population_size, num_jobs, num_machines):
    return [[random.randint(0, num_machines - 1) for _ in
range(num_jobs)] for _ in range(population_size)]

# Calculate makespan
def calculate_makespan(chromosome, job_durations, num_machines):
    machine_times = [0] * num_machines
    for job, machine in enumerate(chromosome):
        machine_times[machine] += job_durations[job]
    return max(machine_times)

# Selection - Tournament selection
def tournament_selection(population, fitness, tournament_size=3):
    selected = []
    for _ in range(len(population)):
        tournament = [random.choice(range(len(population))) for _ in
range(tournament_size)]
        fittest_individual = min(tournament, key=lambda i: fitness[i])
        selected.append(population[fittest_individual])
    return selected

# Crossover - Single point crossover
def crossover(parent1, parent2):
    if random.random() < crossover_rate:
        point = random.randint(1, len(parent1) - 1)
```

```
        return parent1[:point] + parent2[point:], parent2[:point] +
parent1[point:]
    else:
        return parent1, parent2

# Mutation - Randomly change a job's machine assignment
def mutate(chromosome, num_machines, mutation_rate):
    for i in range(len(chromosome)):
        if random.random() < mutation_rate:
            chromosome[i] = random.randint(0, num_machines - 1)
    return chromosome

# Main Genetic Algorithm
population = initialize_population(population_size, num_jobs,
num_machines)

for generation in range(generations):
    # Calculate fitness for each individual
    fitness = [calculate_makespan(individual, job_durations,
num_machines) for individual in population]

    # Selection
    selected = tournament_selection(population, fitness)

    # Crossover
    offspring = []
    for i in range(0, len(selected), 2):
        parent1, parent2 = selected[i], selected[i + 1]
        child1, child2 = crossover(parent1, parent2)
        offspring.extend([child1, child2])

    # Mutation
    population = [mutate(individual, num_machines, mutation_rate) for
individual in offspring]

# Find the best solution
best_solution = min(population, key=lambda chrom:
calculate_makespan(chrom, job_durations, num_machines))
best_makespan = calculate_makespan(best_solution, job_durations,
num_machines)

print("Best Schedule:", best_solution)
print("Best Makespan:", best_makespan)
```

```
Best Schedule: [1, 2, 0, 2, 1]
Best Makespan: 20
```

```
import gym
from gym import spaces
import numpy as np
```

```
import random
from stable_baselines3 import PPO
from stable_baselines3.common.env_util import make_vec_env

import random

# Genetic Algorithm Functions
def initialize_population(population_size, num_jobs, num_machines):
    return [[random.randint(0, num_machines - 1) for _ in
range(num_jobs)] for _ in range(population_size)]

def calculate_makespan(chromosome, job_durations, num_machines):
    machine_times = [0] * num_machines
    for job, machine in enumerate(chromosome):
        machine_times[machine] += job_durations[job]
    return max(machine_times)

def tournament_selection(population, fitness, tournament_size=3):
    selected = []
    for _ in range(len(population)):
        tournament = [random.choice(range(len(population))) for _ in
range(tournament_size)]
        fittest_individual = min(tournament, key=lambda i: fitness[i])
        selected.append(population[fittest_individual])
    return selected

def crossover(parent1, parent2, crossover_rate):
    if random.random() < crossover_rate:
        point = random.randint(1, len(parent1) - 1)
        return parent1[:point] + parent2[point:], parent2[:point] +
parent1[point:]
    else:
        return parent1, parent2

def mutate(chromosome, num_machines, mutation_rate):
    for i in range(len(chromosome)):
        if random.random() < mutation_rate:
            chromosome[i] = random.randint(0, num_machines - 1)
    return chromosome

def run_genetic_algorithm(num_jobs, job_durations, num_machines,
population_size, generations, crossover_rate, mutation_rate):
    population = initialize_population(population_size, num_jobs,
num_machines)

    for generation in range(generations):
        fitness = [calculate_makespan(individual, job_durations,
num_machines) for individual in population]
        selected = tournament_selection(population, fitness)
        offspring = []
```

```
        for i in range(0, len(selected), 2):
            parent1, parent2 = selected[i], selected[i + 1]
            child1, child2 = crossover(parent1, parent2,
crossover_rate)
            offspring.extend([child1, child2])
        population = [mutate(individual, num_machines, mutation_rate)
for individual in offspring]

    best_solution = min(population, key=lambda chrom:
calculate_makespan(chrom, job_durations, num_machines))
    best_makespan = calculate_makespan(best_solution, job_durations,
num_machines)
    print(best_makespan)
    return best_makespan



class JobSchedulingEnv(gym.Env):
    def __init__(self, num_jobs=5, job_durations=[10, 3, 20, 8, 1],
num_machines=3, target_makespan=20,tolerance=1):
        super(JobSchedulingEnv, self).__init__()
        self.num_jobs = num_jobs
        self.job_durations = job_durations
        self.num_machines = num_machines
        self.target_makespan = target_makespan
        self.action_space = spaces.Discrete(num_jobs * num_machines)
# New action space
        self.observation_space = spaces.Box(low=0, high=num_machines,
shape=(num_jobs,), dtype=np.int32)
        self.max_steps = 2000
        self.tolerance = tolerance
        self.state = None
        self.reset()

    def reset(self):
        self.state = np.random.randint(low=0, high=self.num_machines,
size=self.num_jobs)  # Random initial state
        self.current_step = 0
        return self.state

    def step(self, action):
        job_index = action // self.num_machines
        machine_index = action % self.num_machines
        self.state[job_index] = machine_index
        current_makespan = max([sum(self.job_durations[j] for j in
range(self.num_jobs) if self.state[j] == m) for m in
range(self.num_machines)])
        reward = -abs(current_makespan - self.target_makespan)  #
Negative absolute difference as reward
        self.current_step += 1
```

```python
        done = abs(current_makespan - self.target_makespan) <=
self.tolerance
        return self.state, reward, done, {}


    def render(self, mode='human', close=False):
        if close:
            return
        for m in range(self.num_machines):
            jobs_on_machine = [f"J{i+1}" for i in range(self.num_jobs)
if self.state[i] == m]
            makespan = sum(self.job_durations[i] for i in
range(self.num_jobs) if self.state[i] == m)
            print(f"Machine {m+1} - Jobs: {', '.join(jobs_on_machine)}
| Makespan: {makespan} minutes")
        print("-" * 50)

# Training Parameters
number_of_epochs = 3
timesteps_per_epoch = 40000
num_jobs = 5
num_machines = 3
population_size = 50
generations = 100
crossover_rate = 0.8
mutation_rate = 0.1
tolerance = 5

# Initialize the PPO model
dummy_env = JobSchedulingEnv(num_jobs=num_jobs, job_durations=[1] *
num_jobs, num_machines=num_machines,
target_makespan=1,tolerance=tolerance)
model = PPO("MlpPolicy", dummy_env, learning_rate=0.00025,
n_steps=2000, batch_size=64, gamma=0.99, gae_lambda=0.95,
clip_range=0.2, ent_coef=0.01, verbose=1,
tensorboard_log="./ppo_job_scheduling_tensorboard/")

# Training loop
for epoch in range(number_of_epochs):
    # Generate random job durations for each epoch
    job_durations = [random.randint(1, 5) for _ in range(num_jobs)]

    # Run the Genetic Algorithm to find the target makespan
    target_makespan = run_genetic_algorithm(num_jobs, job_durations,
num_machines, population_size, generations, crossover_rate,
mutation_rate)


    # Create the real environment with new parameters
    real_env = JobSchedulingEnv(num_jobs=num_jobs,
```

```
job_durations=job_durations, num_machines=num_machines,
target_makespan=target_makespan,tolerance=tolerance)
    real_env = make_vec_env(lambda: real_env, n_envs=1)

    # Update the model's environment
    model.set_env(real_env)

    # Train the model
    model.learn(total_timesteps=timesteps_per_epoch)


# Saving the model
model.save("job_scheduling_model")
```

```
Using cuda device
Wrapping the env with a `Monitor` wrapper
Wrapping the env in a DummyVecEnv.
3
Logging to ./ppo_job_scheduling_tensorboard/PPO_290
```

```
/usr/local/lib/python3.10/dist-packages/stable_baselines3/ppo/
ppo.py:155: UserWarning: You have specified a mini-batch size of 64,
but because the `RolloutBuffer` is of size `n_steps * n_envs = 2000`,
after every 31 untruncated mini-batches, there will be a truncated
mini-batch of size 16
We recommend using a `batch_size` that is a factor of `n_steps *
n_envs`.
Info: (n_steps=2000 and n_envs=1)
  warnings.warn(
```

```
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 1.04     |
|    ep_rew_mean     | -2.58    |
| time/              |          |
|    fps             | 684      |
|    iterations      | 1        |
|    time_elapsed    | 2        |
|    total_timesteps | 2000     |
---------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.02        |
|    ep_rew_mean          | -2.41       |
| time/                   |             |
|    fps                  | 553         |
|    iterations           | 2           |
|    time_elapsed         | 7           |
|    total_timesteps      | 4000        |
| train/                  |             |
```

```
|    approx_kl            | 0.018658869 |
|    clip_fraction        | 0.149       |
|    clip_range           | 0.2         |
|    entropy_loss         | -2.7        |
|    explained_variance   | -0.0262     |
|    learning_rate        | 0.00025     |
|    loss                 | 0.536       |
|    n_updates            | 10          |
|    policy_gradient_loss | -0.0304     |
|    value_loss           | 4.86        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.02        |
|    ep_rew_mean          | -2.4        |
| time/                   |             |
|    fps                  | 503         |
|    iterations           | 3           |
|    time_elapsed         | 11          |
|    total_timesteps      | 6000        |
| train/                  |             |
|    approx_kl            | 0.021347404 |
|    clip_fraction        | 0.28        |
|    clip_range           | 0.2         |
|    entropy_loss         | -2.64       |
|    explained_variance   | -0.0314     |
|    learning_rate        | 0.00025     |
|    loss                 | 0.853       |
|    n_updates            | 20          |
|    policy_gradient_loss | -0.0446     |
|    value_loss           | 2.46        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.03        |
|    ep_rew_mean          | -2.29       |
| time/                   |             |
|    fps                  | 492         |
|    iterations           | 4           |
|    time_elapsed         | 16          |
|    total_timesteps      | 8000        |
| train/                  |             |
|    approx_kl            | 0.021778788 |
|    clip_fraction        | 0.298       |
|    clip_range           | 0.2         |
|    entropy_loss         | -2.56       |
|    explained_variance   | 0.0757      |
|    learning_rate        | 0.00025     |
|    loss                 | 0.569       |
```

```
|    n_updates            | 30          |
|    policy_gradient_loss | -0.0519     |
|    value_loss           | 2.37        |
-----------------------------------------
----------------------------------------
| rollout/                |            |
|    ep_len_mean          | 1.01       |
|    ep_rew_mean          | -2.07      |
| time/                   |            |
|    fps                  | 486        |
|    iterations           | 5          |
|    time_elapsed         | 20         |
|    total_timesteps      | 10000      |
| train/                  |            |
|    approx_kl            | 0.01946235 |
|    clip_fraction        | 0.365      |
|    clip_range           | 0.2        |
|    entropy_loss         | -2.44      |
|    explained_variance   | 0.122      |
|    learning_rate        | 0.00025    |
|    loss                 | 0.543      |
|    n_updates            | 40         |
|    policy_gradient_loss | -0.0646    |
|    value_loss           | 1.99       |
----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1           |
|    ep_rew_mean          | -1.98       |
| time/                   |             |
|    fps                  | 474         |
|    iterations           | 6           |
|    time_elapsed         | 25          |
|    total_timesteps      | 12000       |
| train/                  |             |
|    approx_kl            | 0.021550588 |
|    clip_fraction        | 0.39        |
|    clip_range           | 0.2         |
|    entropy_loss         | -2.32       |
|    explained_variance   | 0.147       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.495       |
|    n_updates            | 50          |
|    policy_gradient_loss | -0.0745     |
|    value_loss           | 1.56        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.01        |
```

```
|    ep_rew_mean          | -1.93       |
| time/                   |             |
|    fps                  | 473         |
|    iterations           | 7           |
|    time_elapsed         | 29          |
|    total_timesteps      | 14000       |
| train/                  |             |
|    approx_kl            | 0.028465461 |
|    clip_fraction        | 0.401       |
|    clip_range           | 0.2         |
|    entropy_loss         | -2.16       |
|    explained_variance   | 0.13        |
|    learning_rate        | 0.00025     |
|    loss                 | 0.183       |
|    n_updates            | 60          |
|    policy_gradient_loss | -0.0762     |
|    value_loss           | 1.15        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1           |
|    ep_rew_mean          | -1.63       |
| time/                   |             |
|    fps                  | 468         |
|    iterations           | 8           |
|    time_elapsed         | 34          |
|    total_timesteps      | 16000       |
| train/                  |             |
|    approx_kl            | 0.020976381 |
|    clip_fraction        | 0.261       |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.99       |
|    explained_variance   | 0.0626      |
|    learning_rate        | 0.00025     |
|    loss                 | 0.327       |
|    n_updates            | 70          |
|    policy_gradient_loss | -0.0611     |
|    value_loss           | 0.928       |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1           |
|    ep_rew_mean          | -1.5        |
| time/                   |             |
|    fps                  | 465         |
|    iterations           | 9           |
|    time_elapsed         | 38          |
|    total_timesteps      | 18000       |
| train/                  |             |
```

```
|    approx_kl            | 0.018754754 |
|    clip_fraction        | 0.222       |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.85       |
|    explained_variance   | 0.0648      |
|    learning_rate        | 0.00025     |
|    loss                 | 0.261       |
|    n_updates            | 80          |
|    policy_gradient_loss | -0.0523     |
|    value_loss           | 0.8         |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1           |
|    ep_rew_mean          | -1.49       |
| time/                   |             |
|    fps                  | 465         |
|    iterations           | 10          |
|    time_elapsed         | 43          |
|    total_timesteps      | 20000       |
| train/                  |             |
|    approx_kl            | 0.016115401 |
|    clip_fraction        | 0.213       |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.71       |
|    explained_variance   | 0.105       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.198       |
|    n_updates            | 90          |
|    policy_gradient_loss | -0.0449     |
|    value_loss           | 0.614       |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1           |
|    ep_rew_mean          | -1.36       |
| time/                   |             |
|    fps                  | 461         |
|    iterations           | 11          |
|    time_elapsed         | 47          |
|    total_timesteps      | 22000       |
| train/                  |             |
|    approx_kl            | 0.015000742 |
|    clip_fraction        | 0.145       |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.63       |
|    explained_variance   | 0.0638      |
|    learning_rate        | 0.00025     |
|    loss                 | 0.138       |
```

```
|    n_updates             | 100         |
|    policy_gradient_loss | -0.0382     |
|    value_loss            | 0.688       |
-----------------------------------------
-----------------------------------------
| rollout/                 |             |
|    ep_len_mean           | 1           |
|    ep_rew_mean           | -1.43       |
| time/                    |             |
|    fps                   | 461         |
|    iterations            | 12          |
|    time_elapsed          | 52          |
|    total_timesteps       | 24000       |
| train/                   |             |
|    approx_kl             | 0.013525712 |
|    clip_fraction         | 0.143       |
|    clip_range            | 0.2         |
|    entropy_loss          | -1.54       |
|    explained_variance    | 0.101       |
|    learning_rate         | 0.00025     |
|    loss                  | 0.206       |
|    n_updates             | 110         |
|    policy_gradient_loss | -0.0371     |
|    value_loss            | 0.566       |
-----------------------------------------
-----------------------------------------
| rollout/                 |             |
|    ep_len_mean           | 1           |
|    ep_rew_mean           | -1.43       |
| time/                    |             |
|    fps                   | 460         |
|    iterations            | 13          |
|    time_elapsed          | 56          |
|    total_timesteps       | 26000       |
| train/                   |             |
|    approx_kl             | 0.015544775 |
|    clip_fraction         | 0.165       |
|    clip_range            | 0.2         |
|    entropy_loss          | -1.43       |
|    explained_variance    | 0.0905      |
|    learning_rate         | 0.00025     |
|    loss                  | 0.168       |
|    n_updates             | 120         |
|    policy_gradient_loss | -0.0383     |
|    value_loss            | 0.523       |
-----------------------------------------
-----------------------------------------
| rollout/                 |             |
|    ep_len_mean           | 1           |
```

```
|    ep_rew_mean          | -1.3        |
| time/                   |             |
|    fps                  | 457         |
|    iterations           | 14          |
|    time_elapsed         | 61          |
|    total_timesteps      | 28000       |
| train/                  |             |
|    approx_kl            | 0.014967583 |
|    clip_fraction        | 0.155       |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.38       |
|    explained_variance   | 0.118       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.211       |
|    n_updates            | 130         |
|    policy_gradient_loss | -0.0362     |
|    value_loss           | 0.516       |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1           |
|    ep_rew_mean          | -1.22       |
| time/                   |             |
|    fps                  | 457         |
|    iterations           | 15          |
|    time_elapsed         | 65          |
|    total_timesteps      | 30000       |
| train/                  |             |
|    approx_kl            | 0.013354813 |
|    clip_fraction        | 0.112       |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.29       |
|    explained_variance   | 0.0782      |
|    learning_rate        | 0.00025     |
|    loss                 | 0.228       |
|    n_updates            | 140         |
|    policy_gradient_loss | -0.0335     |
|    value_loss           | 0.499       |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1           |
|    ep_rew_mean          | -1.45       |
| time/                   |             |
|    fps                  | 455         |
|    iterations           | 16          |
|    time_elapsed         | 70          |
|    total_timesteps      | 32000       |
| train/                  |             |
```

```
|    approx_kl            | 0.010692762 |
|    clip_fraction        | 0.102       |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.22       |
|    explained_variance   | 0.0932      |
|    learning_rate        | 0.00025     |
|    loss                 | 0.278       |
|    n_updates            | 150         |
|    policy_gradient_loss | -0.0306     |
|    value_loss           | 0.484       |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1           |
|    ep_rew_mean          | -1.29       |
| time/                   |             |
|    fps                  | 455         |
|    iterations           | 17          |
|    time_elapsed         | 74          |
|    total_timesteps      | 34000       |
| train/                  |             |
|    approx_kl            | 0.014578375 |
|    clip_fraction        | 0.0936      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.17       |
|    explained_variance   | 0.11        |
|    learning_rate        | 0.00025     |
|    loss                 | 0.431       |
|    n_updates            | 160         |
|    policy_gradient_loss | -0.0256     |
|    value_loss           | 0.441       |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1           |
|    ep_rew_mean          | -1.38       |
| time/                   |             |
|    fps                  | 455         |
|    iterations           | 18          |
|    time_elapsed         | 78          |
|    total_timesteps      | 36000       |
| train/                  |             |
|    approx_kl            | 0.011278259 |
|    clip_fraction        | 0.13        |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.13       |
|    explained_variance   | 0.129       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.131       |
```

```
|    n_updates            | 170         |
|    policy_gradient_loss | -0.0309     |
|    value_loss           | 0.463       |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1           |
|    ep_rew_mean          | -1.19       |
| time/                   |             |
|    fps                  | 453         |
|    iterations           | 19          |
|    time_elapsed         | 83          |
|    total_timesteps      | 38000       |
| train/                  |             |
|    approx_kl            | 0.009060815 |
|    clip_fraction        | 0.0945      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.06       |
|    explained_variance   | 0.11        |
|    learning_rate        | 0.00025     |
|    loss                 | 0.136       |
|    n_updates            | 180         |
|    policy_gradient_loss | -0.0258     |
|    value_loss           | 0.442       |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1           |
|    ep_rew_mean          | -1.34       |
| time/                   |             |
|    fps                  | 453         |
|    iterations           | 20          |
|    time_elapsed         | 88          |
|    total_timesteps      | 40000       |
| train/                  |             |
|    approx_kl            | 0.008264336 |
|    clip_fraction        | 0.0913      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1          |
|    explained_variance   | 0.178       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.158       |
|    n_updates            | 190         |
|    policy_gradient_loss | -0.0216     |
|    value_loss           | 0.399       |
-----------------------------------------
7
Logging to ./ppo_job_scheduling_tensorboard/PPO_291
---------------------------------
```

```
| rollout/           |          |
|    ep_len_mean     | 1.03     |
|    ep_rew_mean     | -2.31    |
| time/              |          |
|    fps             | 664      |
|    iterations      | 1        |
|    time_elapsed    | 3        |
|    total_timesteps | 2000     |
---------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.08        |
|    ep_rew_mean          | -2.75       |
| time/                   |             |
|    fps                  | 527         |
|    iterations           | 2           |
|    time_elapsed         | 7           |
|    total_timesteps      | 4000        |
| train/                  |             |
|    approx_kl            | 0.010680847 |
|    clip_fraction        | 0.0947      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.934      |
|    explained_variance   | 0.0364      |
|    learning_rate        | 0.00025     |
|    loss                 | 1.87        |
|    n_updates            | 210         |
|    policy_gradient_loss | -0.019      |
|    value_loss           | 5.42        |
-----------------------------------------
----------------------------------------
| rollout/                |            |
|    ep_len_mean          | 1.01       |
|    ep_rew_mean          | -2.13      |
| time/                   |            |
|    fps                  | 507        |
|    iterations           | 3          |
|    time_elapsed         | 11         |
|    total_timesteps      | 6000       |
| train/                  |            |
|    approx_kl            | 0.00815574 |
|    clip_fraction        | 0.0967     |
|    clip_range           | 0.2        |
|    entropy_loss         | -0.921     |
|    explained_variance   | 0.121      |
|    learning_rate        | 0.00025    |
|    loss                 | 0.665      |
|    n_updates            | 220        |
|    policy_gradient_loss | -0.0183    |
```

```
|    value_loss           | 4.63        |
-----------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 1.03         |
|    ep_rew_mean          | -2.48        |
| time/                   |              |
|    fps                  | 485          |
|    iterations           | 4            |
|    time_elapsed         | 16           |
|    total_timesteps      | 8000         |
| train/                  |              |
|    approx_kl            | 0.010172943  |
|    clip_fraction        | 0.102        |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.898       |
|    explained_variance   | 0.208        |
|    learning_rate        | 0.00025      |
|    loss                 | 0.895        |
|    n_updates            | 230          |
|    policy_gradient_loss | -0.0221      |
|    value_loss           | 4.27         |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 1.01         |
|    ep_rew_mean          | -2.01        |
| time/                   |              |
|    fps                  | 482          |
|    iterations           | 5            |
|    time_elapsed         | 20           |
|    total_timesteps      | 10000        |
| train/                  |              |
|    approx_kl            | 0.011878994  |
|    clip_fraction        | 0.138        |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.914       |
|    explained_variance   | 0.271        |
|    learning_rate        | 0.00025      |
|    loss                 | 0.992        |
|    n_updates            | 240          |
|    policy_gradient_loss | -0.0256      |
|    value_loss           | 3.81         |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 1            |
|    ep_rew_mean          | -2.1         |
| time/                   |              |
|    fps                  | 478          |
```

```
|    iterations            | 6           |
|    time_elapsed          | 25          |
|    total_timesteps       | 12000       |
| train/                   |             |
|    approx_kl             | 0.009134321 |
|    clip_fraction         | 0.104       |
|    clip_range            | 0.2         |
|    entropy_loss          | -0.866      |
|    explained_variance    | 0.288       |
|    learning_rate         | 0.00025     |
|    loss                  | 0.91        |
|    n_updates             | 250         |
|    policy_gradient_loss  | -0.0221     |
|    value_loss            | 2.96        |
-----------------------------------------
-----------------------------------------
| rollout/                 |             |
|    ep_len_mean           | 1           |
|    ep_rew_mean           | -2.01       |
| time/                    |             |
|    fps                   | 469         |
|    iterations            | 7           |
|    time_elapsed          | 29          |
|    total_timesteps       | 14000       |
| train/                   |             |
|    approx_kl             | 0.011752756 |
|    clip_fraction         | 0.122       |
|    clip_range            | 0.2         |
|    entropy_loss          | -0.864      |
|    explained_variance    | 0.294       |
|    learning_rate         | 0.00025     |
|    loss                  | 3.14        |
|    n_updates             | 260         |
|    policy_gradient_loss  | -0.0243     |
|    value_loss            | 2.87        |
-----------------------------------------
-----------------------------------------
| rollout/                 |             |
|    ep_len_mean           | 1.01        |
|    ep_rew_mean           | -1.9        |
| time/                    |             |
|    fps                   | 469         |
|    iterations            | 8           |
|    time_elapsed          | 34          |
|    total_timesteps       | 16000       |
| train/                   |             |
|    approx_kl             | 0.008926092 |
|    clip_fraction         | 0.0964      |
|    clip_range            | 0.2         |
```

```
|    entropy_loss         | -0.852      |
|    explained_variance   | 0.357       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.354       |
|    n_updates            | 270         |
|    policy_gradient_loss | -0.0218     |
|    value_loss           | 2.35        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1           |
|    ep_rew_mean          | -1.97       |
| time/                   |             |
|    fps                  | 467         |
|    iterations           | 9           |
|    time_elapsed         | 38          |
|    total_timesteps      | 18000       |
| train/                  |             |
|    approx_kl            | 0.009041598 |
|    clip_fraction        | 0.0971      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.842      |
|    explained_variance   | 0.342       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.87        |
|    n_updates            | 280         |
|    policy_gradient_loss | -0.0203     |
|    value_loss           | 2.14        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1           |
|    ep_rew_mean          | -1.59       |
| time/                   |             |
|    fps                  | 464         |
|    iterations           | 10          |
|    time_elapsed         | 43          |
|    total_timesteps      | 20000       |
| train/                  |             |
|    approx_kl            | 0.010144679 |
|    clip_fraction        | 0.118       |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.831      |
|    explained_variance   | 0.45        |
|    learning_rate        | 0.00025     |
|    loss                 | 1.06        |
|    n_updates            | 290         |
|    policy_gradient_loss | -0.0216     |
|    value_loss           | 1.8         |
```

```
-----------------------------------------
----------------------------------------
| rollout/                |            |
|    ep_len_mean          | 1.01       |
|    ep_rew_mean          | -2.06      |
| time/                   |            |
|    fps                  | 464        |
|    iterations           | 11         |
|    time_elapsed         | 47         |
|    total_timesteps      | 22000      |
| train/                  |            |
|    approx_kl            | 0.00745301 |
|    clip_fraction        | 0.104      |
|    clip_range           | 0.2        |
|    entropy_loss         | -0.829     |
|    explained_variance   | 0.424      |
|    learning_rate        | 0.00025    |
|    loss                 | 1.2        |
|    n_updates            | 300        |
|    policy_gradient_loss | -0.0233    |
|    value_loss           | 1.85       |
----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.04        |
|    ep_rew_mean          | -2.17       |
| time/                   |             |
|    fps                  | 459         |
|    iterations           | 12          |
|    time_elapsed         | 52          |
|    total_timesteps      | 24000       |
| train/                  |             |
|    approx_kl            | 0.006748994 |
|    clip_fraction        | 0.09        |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.809      |
|    explained_variance   | 0.458       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.254       |
|    n_updates            | 310         |
|    policy_gradient_loss | -0.0245     |
|    value_loss           | 1.62        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.03        |
|    ep_rew_mean          | -2.26       |
| time/                   |             |
|    fps                  | 460         |
```

```
|    iterations           | 13          |
|    time_elapsed         | 56          |
|    total_timesteps      | 26000       |
| train/                  |             |
|    approx_kl            | 0.008977774 |
|    clip_fraction        | 0.135       |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.807      |
|    explained_variance   | 0.449       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.316       |
|    n_updates            | 320         |
|    policy_gradient_loss | -0.0283     |
|    value_loss           | 1.73        |
-----------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 1.04         |
|    ep_rew_mean          | -2.11        |
| time/                   |              |
|    fps                  | 460          |
|    iterations           | 14           |
|    time_elapsed         | 60           |
|    total_timesteps      | 28000        |
| train/                  |              |
|    approx_kl            | 0.0083325375 |
|    clip_fraction        | 0.0859       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.732       |
|    explained_variance   | 0.508        |
|    learning_rate        | 0.00025      |
|    loss                 | 0.586        |
|    n_updates            | 330          |
|    policy_gradient_loss | -0.0241      |
|    value_loss           | 1.49         |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 1            |
|    ep_rew_mean          | -1.74        |
| time/                   |              |
|    fps                  | 458          |
|    iterations           | 15           |
|    time_elapsed         | 65           |
|    total_timesteps      | 30000        |
| train/                  |              |
|    approx_kl            | 0.0094406605 |
|    clip_fraction        | 0.109        |
|    clip_range           | 0.2          |
```

```
|    entropy_loss         | -0.726     |
|    explained_variance   | 0.536      |
|    learning_rate        | 0.00025    |
|    loss                 | 1.17       |
|    n_updates            | 340        |
|    policy_gradient_loss | -0.0258    |
|    value_loss           | 1.25       |
-----------------------------------------
-----------------------------------------
| rollout/                |            |
|    ep_len_mean          | 1.01       |
|    ep_rew_mean          | -1.87      |
| time/                   |            |
|    fps                  | 458        |
|    iterations           | 16         |
|    time_elapsed         | 69         |
|    total_timesteps      | 32000      |
| train/                  |            |
|    approx_kl            | 0.008126431 |
|    clip_fraction        | 0.0842     |
|    clip_range           | 0.2        |
|    entropy_loss         | -0.682     |
|    explained_variance   | 0.579      |
|    learning_rate        | 0.00025    |
|    loss                 | 0.133      |
|    n_updates            | 350        |
|    policy_gradient_loss | -0.0227    |
|    value_loss           | 1.16       |
-----------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 1.01         |
|    ep_rew_mean          | -1.87        |
| time/                   |              |
|    fps                  | 457          |
|    iterations           | 17           |
|    time_elapsed         | 74           |
|    total_timesteps      | 34000        |
| train/                  |              |
|    approx_kl            | 0.0076401755 |
|    clip_fraction        | 0.0826       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.638       |
|    explained_variance   | 0.617        |
|    learning_rate        | 0.00025      |
|    loss                 | 0.664        |
|    n_updates            | 360          |
|    policy_gradient_loss | -0.0202      |
|    value_loss           | 1.09         |
```

```
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.03        |
|    ep_rew_mean          | -1.96       |
| time/                   |             |
|    fps                  | 457         |
|    iterations           | 18          |
|    time_elapsed         | 78          |
|    total_timesteps      | 36000       |
| train/                  |             |
|    approx_kl            | 0.0066925795 |
|    clip_fraction        | 0.0892      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.633      |
|    explained_variance   | 0.648       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.64        |
|    n_updates            | 370         |
|    policy_gradient_loss | -0.0218     |
|    value_loss           | 0.991       |
-----------------------------------------
----------------------------------------
| rollout/                |            |
|    ep_len_mean          | 1.03       |
|    ep_rew_mean          | -1.82      |
| time/                   |            |
|    fps                  | 457        |
|    iterations           | 19         |
|    time_elapsed         | 83         |
|    total_timesteps      | 38000      |
| train/                  |            |
|    approx_kl            | 0.008884434 |
|    clip_fraction        | 0.0955     |
|    clip_range           | 0.2        |
|    entropy_loss         | -0.609     |
|    explained_variance   | 0.646      |
|    learning_rate        | 0.00025    |
|    loss                 | 0.415      |
|    n_updates            | 380        |
|    policy_gradient_loss | -0.0227    |
|    value_loss           | 0.936      |
----------------------------------------
----------------------------------------
| rollout/                |            |
|    ep_len_mean          | 1.01       |
|    ep_rew_mean          | -1.7       |
| time/                   |            |
|    fps                  | 455        |
```

```
|    iterations           | 20          |
|    time_elapsed         | 87          |
|    total_timesteps      | 40000       |
| train/                  |             |
|    approx_kl            | 0.014576204 |
|    clip_fraction        | 0.108       |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.596      |
|    explained_variance   | 0.648       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.0926      |
|    n_updates            | 390         |
|    policy_gradient_loss | -0.0234     |
|    value_loss           | 0.863       |
-----------------------------------------
8
Logging to ./ppo_job_scheduling_tensorboard/PPO_292
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 1.05     |
|    ep_rew_mean     | -3.09    |
| time/              |          |
|    fps             | 692      |
|    iterations      | 1        |
|    time_elapsed    | 2        |
|    total_timesteps | 2000     |
---------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.08        |
|    ep_rew_mean          | -2.82       |
| time/                   |             |
|    fps                  | 558         |
|    iterations           | 2           |
|    time_elapsed         | 7           |
|    total_timesteps      | 4000        |
| train/                  |             |
|    approx_kl            | 0.006262525 |
|    clip_fraction        | 0.062       |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.565      |
|    explained_variance   | 0.428       |
|    learning_rate        | 0.00025     |
|    loss                 | 5.46        |
|    n_updates            | 410         |
|    policy_gradient_loss | -0.0182     |
|    value_loss           | 3.5         |
-----------------------------------------
----------------------------------------
```

```
| rollout/                |            |
|    ep_len_mean          | 1.04       |
|    ep_rew_mean          | -2.84      |
| time/                   |            |
|    fps                  | 506        |
|    iterations           | 3          |
|    time_elapsed         | 11         |
|    total_timesteps      | 6000       |
| train/                  |            |
|    approx_kl            | 0.00887173 |
|    clip_fraction        | 0.092      |
|    clip_range           | 0.2        |
|    entropy_loss         | -0.587     |
|    explained_variance   | 0.568      |
|    learning_rate        | 0.00025    |
|    loss                 | 0.619      |
|    n_updates            | 420        |
|    policy_gradient_loss | -0.0193    |
|    value_loss           | 2.28       |
----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.02        |
|    ep_rew_mean          | -2.35       |
| time/                   |             |
|    fps                  | 495         |
|    iterations           | 4           |
|    time_elapsed         | 16          |
|    total_timesteps      | 8000        |
| train/                  |             |
|    approx_kl            | 0.011628484 |
|    clip_fraction        | 0.0972      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.57       |
|    explained_variance   | 0.656       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.444       |
|    n_updates            | 430         |
|    policy_gradient_loss | -0.0253     |
|    value_loss           | 2.36        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.01        |
|    ep_rew_mean          | -2.45       |
| time/                   |             |
|    fps                  | 483         |
|    iterations           | 5           |
|    time_elapsed         | 20          |
```

```
|    total_timesteps      | 10000       |
| train/                  |             |
|    approx_kl            | 0.009056151 |
|    clip_fraction        | 0.0908      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.568      |
|    explained_variance   | 0.694       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.414       |
|    n_updates            | 440         |
|    policy_gradient_loss | -0.0251     |
|    value_loss           | 1.47        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.08        |
|    ep_rew_mean          | -3.1        |
| time/                   |             |
|    fps                  | 479         |
|    iterations           | 6           |
|    time_elapsed         | 25          |
|    total_timesteps      | 12000       |
| train/                  |             |
|    approx_kl            | 0.008488921 |
|    clip_fraction        | 0.0963      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.599      |
|    explained_variance   | 0.698       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.41        |
|    n_updates            | 450         |
|    policy_gradient_loss | -0.0279     |
|    value_loss           | 1.7         |
-----------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 1.03         |
|    ep_rew_mean          | -2.56        |
| time/                   |              |
|    fps                  | 477          |
|    iterations           | 7            |
|    time_elapsed         | 29           |
|    total_timesteps      | 14000        |
| train/                  |              |
|    approx_kl            | 0.0091473125 |
|    clip_fraction        | 0.0952       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.599       |
|    explained_variance   | 0.659        |
```

```
|    learning_rate        | 0.00025      |
|    loss                 | 0.93         |
|    n_updates            | 460          |
|    policy_gradient_loss | -0.0293      |
|    value_loss           | 1.83         |
------------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.06        |
|    ep_rew_mean          | -2.79       |
| time/                   |             |
|    fps                  | 469         |
|    iterations           | 8           |
|    time_elapsed         | 34          |
|    total_timesteps      | 16000       |
| train/                  |             |
|    approx_kl            | 0.012551058 |
|    clip_fraction        | 0.13        |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.59       |
|    explained_variance   | 0.708       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.932       |
|    n_updates            | 470         |
|    policy_gradient_loss | -0.0333     |
|    value_loss           | 1.51        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.04        |
|    ep_rew_mean          | -2.74       |
| time/                   |             |
|    fps                  | 470         |
|    iterations           | 9           |
|    time_elapsed         | 38          |
|    total_timesteps      | 18000       |
| train/                  |             |
|    approx_kl            | 0.010959093 |
|    clip_fraction        | 0.101       |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.585      |
|    explained_variance   | 0.707       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.745       |
|    n_updates            | 480         |
|    policy_gradient_loss | -0.0261     |
|    value_loss           | 1.52        |
-----------------------------------------
-----------------------------------------
```

```
| rollout/                |             |
|    ep_len_mean          | 1.02        |
|    ep_rew_mean          | -2.46       |
| time/                   |             |
|    fps                  | 469         |
|    iterations           | 10          |
|    time_elapsed         | 42          |
|    total_timesteps      | 20000       |
| train/                  |             |
|    approx_kl            | 0.013288228 |
|    clip_fraction        | 0.13        |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.608      |
|    explained_variance   | 0.718       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.24        |
|    n_updates            | 490         |
|    policy_gradient_loss | -0.033      |
|    value_loss           | 1.4         |
-----------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 1.03         |
|    ep_rew_mean          | -2.23        |
| time/                   |              |
|    fps                  | 465          |
|    iterations           | 11           |
|    time_elapsed         | 47           |
|    total_timesteps      | 22000        |
| train/                  |              |
|    approx_kl            | 0.0113274045 |
|    clip_fraction        | 0.11         |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.607       |
|    explained_variance   | 0.756        |
|    learning_rate        | 0.00025      |
|    loss                 | 0.658        |
|    n_updates            | 500          |
|    policy_gradient_loss | -0.03        |
|    value_loss           | 1.25         |
------------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.03        |
|    ep_rew_mean          | -2.43       |
| time/                   |             |
|    fps                  | 465         |
|    iterations           | 12          |
|    time_elapsed         | 51          |
|    total_timesteps      | 24000       |
```

```
| train/                  |             |
|    approx_kl            | 0.010241794 |
|    clip_fraction        | 0.128       |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.61       |
|    explained_variance   | 0.701       |
|    learning_rate        | 0.00025     |
|    loss                 | 1.29        |
|    n_updates            | 510         |
|    policy_gradient_loss | -0.0337     |
|    value_loss           | 1.37        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.03        |
|    ep_rew_mean          | -2.37       |
| time/                   |             |
|    fps                  | 461         |
|    iterations           | 13          |
|    time_elapsed         | 56          |
|    total_timesteps      | 26000       |
| train/                  |             |
|    approx_kl            | 0.013626059 |
|    clip_fraction        | 0.127       |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.579      |
|    explained_variance   | 0.682       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.699       |
|    n_updates            | 520         |
|    policy_gradient_loss | -0.0339     |
|    value_loss           | 1.64        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1           |
|    ep_rew_mean          | -2.14       |
| time/                   |             |
|    fps                  | 460         |
|    iterations           | 14          |
|    time_elapsed         | 60          |
|    total_timesteps      | 28000       |
| train/                  |             |
|    approx_kl            | 0.010502832 |
|    clip_fraction        | 0.108       |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.569      |
|    explained_variance   | 0.71        |
|    learning_rate        | 0.00025     |
```

```
|    loss                 | 0.141       |
|    n_updates            | 530         |
|    policy_gradient_loss | -0.03       |
|    value_loss           | 1.52        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.04        |
|    ep_rew_mean          | -2.54       |
| time/                   |             |
|    fps                  | 461         |
|    iterations           | 15          |
|    time_elapsed         | 65          |
|    total_timesteps      | 30000       |
| train/                  |             |
|    approx_kl            | 0.009579731 |
|    clip_fraction        | 0.107       |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.583      |
|    explained_variance   | 0.744       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.125       |
|    n_updates            | 540         |
|    policy_gradient_loss | -0.0325     |
|    value_loss           | 1.13        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.01        |
|    ep_rew_mean          | -2.15       |
| time/                   |             |
|    fps                  | 458         |
|    iterations           | 16          |
|    time_elapsed         | 69          |
|    total_timesteps      | 32000       |
| train/                  |             |
|    approx_kl            | 0.007946995 |
|    clip_fraction        | 0.0918      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.566      |
|    explained_variance   | 0.736       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.224       |
|    n_updates            | 550         |
|    policy_gradient_loss | -0.0323     |
|    value_loss           | 1.19        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
```

```
|    ep_len_mean          | 1.01        |
|    ep_rew_mean          | -1.79       |
| time/                   |             |
|    fps                  | 458         |
|    iterations           | 17          |
|    time_elapsed         | 74          |
|    total_timesteps      | 34000       |
| train/                  |             |
|    approx_kl            | 0.008344554 |
|    clip_fraction        | 0.0919      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.548      |
|    explained_variance   | 0.75        |
|    learning_rate        | 0.00025     |
|    loss                 | 0.306       |
|    n_updates            | 560         |
|    policy_gradient_loss | -0.0283     |
|    value_loss           | 1.22        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1           |
|    ep_rew_mean          | -1.87       |
| time/                   |             |
|    fps                  | 459         |
|    iterations           | 18          |
|    time_elapsed         | 78          |
|    total_timesteps      | 36000       |
| train/                  |             |
|    approx_kl            | 0.009061569 |
|    clip_fraction        | 0.0885      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.536      |
|    explained_variance   | 0.726       |
|    learning_rate        | 0.00025     |
|    loss                 | 1.03        |
|    n_updates            | 570         |
|    policy_gradient_loss | -0.0279     |
|    value_loss           | 1.3         |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1.02        |
|    ep_rew_mean          | -1.9        |
| time/                   |             |
|    fps                  | 456         |
|    iterations           | 19          |
|    time_elapsed         | 83          |
|    total_timesteps      | 38000       |
```

```
| train/                  |              |
|    approx_kl            | 0.007086073 |
|    clip_fraction        | 0.0761      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.533      |
|    explained_variance   | 0.753       |
|    learning_rate        | 0.00025     |
|    loss                 | 0.175       |
|    n_updates            | 580         |
|    policy_gradient_loss | -0.0283     |
|    value_loss           | 1.17        |
-----------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 1.02         |
|    ep_rew_mean          | -1.96        |
| time/                   |              |
|    fps                  | 457          |
|    iterations           | 20           |
|    time_elapsed         | 87           |
|    total_timesteps      | 40000        |
| train/                  |              |
|    approx_kl            | 0.0077676857 |
|    clip_fraction        | 0.0776       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.522       |
|    explained_variance   | 0.76         |
|    learning_rate        | 0.00025      |
|    loss                 | 0.254        |
|    n_updates            | 590          |
|    policy_gradient_loss | -0.0268      |
|    value_loss           | 1.18         |
------------------------------------------
```

```
from stable_baselines3 import PPO

# Load the trained model
model = PPO.load("job_scheduling_model")

# Test job durations and environment setup
test_job_durations = [1, 2, 3, 5, 1]  # Example job durations
num_jobs = len(test_job_durations)
num_machines = 3
test_env = JobSchedulingEnv(num_jobs=num_jobs,
job_durations=test_job_durations, num_machines=num_machines,
target_makespan=20, tolerance=1)

# Run the model in the test environment
obs = test_env.reset()
done = False
```

```
max_iterations = 30000  # Prevent infinite loop
iteration = 0

while not done and iteration < max_iterations:
    action, _states = model.predict(obs, deterministic=True)
    obs, rewards, done, info = test_env.step(action)
    iteration += 1


# Check if loop exited due to reaching max iterations
if iteration >= max_iterations:
    print("Reached maximum iterations without fulfilling termination
conditions.")

# Print the schedule
def print_schedule(env):
    print("\nOptimal Schedule:")
    for m in range(env.num_machines):
        jobs_on_machine = [f"J{i+1}" for i in range(env.num_jobs) if
env.state[i] == m]
        makespan = sum(env.job_durations[i] for i in
range(env.num_jobs) if env.state[i] == m)
        print(f"Machine {m+1} - Jobs: {', '.join(jobs_on_machine)} |
Makespan: {makespan} minutes")

print_schedule(test_env)
```

```
Reached maximum iterations without fulfilling termination conditions.

Optimal Schedule:
Machine 1 - Jobs: J5 | Makespan: 1 minutes
Machine 2 - Jobs: J1, J3 | Makespan: 4 minutes
Machine 3 - Jobs: J2, J4 | Makespan: 7 minutes
```

```
import random

# Parameters
num_jobs = 10
job_durations = [10, 3, 20, 8, 1,10, 3, 20, 8, 1]
num_machines = 5
population_size = 50
generations = 100
crossover_rate = 0.8
mutation_rate = 0.1

# Initialize population
def initialize_population(population_size, num_jobs, num_machines):
    return [[random.randint(0, num_machines - 1) for _ in
range(num_jobs)] for _ in range(population_size)]

# Calculate makespan and balance
```

```python
def calculate_makespan_and_balance(chromosome, job_durations,
num_machines):
    machine_times = [0] * num_machines
    for job, machine in enumerate(chromosome):
        machine_times[machine] += job_durations[job]
    max_makespan = max(machine_times)
    balance_penalty = sum([(max_makespan - time)**2 for time in
machine_times])  # Penalize unbalanced schedules
    return max_makespan + balance_penalty

# Tournament selection
def tournament_selection(population, fitness, tournament_size=3):
    selected = []
    for _ in range(len(population)):
        tournament = [random.choice(range(len(population))) for _ in
range(tournament_size)]
        fittest_individual = min(tournament, key=lambda i: fitness[i])
        selected.append(population[fittest_individual])
    return selected

# Crossover - Single point crossover
def crossover(parent1, parent2):
    if random.random() < crossover_rate:
        point = random.randint(1, len(parent1) - 1)
        return parent1[:point] + parent2[point:], parent2[:point] +
parent1[point:]
    else:
        return parent1, parent2

# Mutation - Randomly change a job's machine assignment
def mutate(chromosome, num_machines, mutation_rate):
    for i in range(len(chromosome)):
        if random.random() < mutation_rate:
            chromosome[i] = random.randint(0, num_machines - 1)
    return chromosome

# Function to create a readable schedule from the chromosome
def create_schedule(chromosome, job_durations):
    schedule = {machine: [] for machine in range(num_machines)}
    for job, machine in enumerate(chromosome):
        schedule[machine].append((f"J{job+1}", job_durations[job]))
    return schedule

# Main Genetic Algorithm
population = initialize_population(population_size, num_jobs,
num_machines)

for generation in range(generations):
    fitness = [calculate_makespan_and_balance(individual,
job_durations, num_machines) for individual in population]
```

```
    selected = tournament_selection(population, fitness)
    offspring = []
    for i in range(0, len(selected), 2):
        parent1, parent2 = selected[i], selected[i + 1]
        child1, child2 = crossover(parent1, parent2)
        offspring.extend([child1, child2])
    population = [mutate(individual, num_machines, mutation_rate) for
individual in offspring]

# Find the best solution and create schedule
best_solution = min(population, key=lambda chrom:
calculate_makespan_and_balance(chrom, job_durations, num_machines))
best_schedule = create_schedule(best_solution, job_durations)
print(best_schedule)
# Displaying the schedule
print("Optimal Schedule:")
for machine, jobs in best_schedule.items():
    job_list = ', '.join([job[0] for job in jobs])
    makespan = sum([job[1] for job in jobs])
    print(f"Machine {machine + 1} - Jobs: {job_list} | Makespan:
{makespan} minutes")

{0: [('J4', 8), ('J9', 8)], 1: [('J1', 10), ('J5', 1), ('J7', 3)], 2:
[('J2', 3), ('J6', 10), ('J10', 1)], 3: [('J8', 20)], 4: [('J3', 20)]}
Optimal Schedule:
Machine 1 - Jobs: J4, J9 | Makespan: 16 minutes
Machine 2 - Jobs: J1, J5, J7 | Makespan: 14 minutes
Machine 3 - Jobs: J2, J6, J10 | Makespan: 14 minutes
Machine 4 - Jobs: J8 | Makespan: 20 minutes
Machine 5 - Jobs: J3 | Makespan: 20 minutes

# Revised approach to include the robot cell information in the
schedule

# Original schedule
original_schedule = best_schedule

# User input for robot cells
user_input = {
    "R 1": [1, 3],
    "R 2": [2,5],
    "R 3": [4]

}

# Function to find the robot cell for a given machine
def find_robot_cell(machine_number, user_input):
    for cell_name, machines in user_input.items():
        if machine_number in machines:
            return cell_name
```

```
    return None

# Adding robot cell information to each schedule
for machine, jobs in original_schedule.items():
    robot_cell = find_robot_cell(machine + 1, user_input)  # +1
because machine numbering starts from 1
    original_schedule[machine] = (robot_cell, jobs)

# Re-arranging the schedule by robot cell
rearranged_schedule = dict(sorted(original_schedule.items(),
key=lambda item: item[1][0]))

# Displaying the rearranged schedule
for machine, (cell, jobs) in rearranged_schedule.items():
    job_list = ', '.join([job[0] for job in jobs])
    makespan = sum([job[1] for job in jobs])
    print(f"Machine {machine} (in {cell}) - Jobs: {job_list} |
Makespan: {makespan} minutes")

# Return rearranged_schedule for further analysis if needed
rearranged_schedule
```

```
Machine 0 (in R 1) - Jobs: J4, J9 | Makespan: 16 minutes
Machine 2 (in R 1) - Jobs: J2, J6, J10 | Makespan: 14 minutes
Machine 1 (in R 2) - Jobs: J1, J5, J7 | Makespan: 14 minutes
Machine 4 (in R 2) - Jobs: J3 | Makespan: 20 minutes
Machine 3 (in R 3) - Jobs: J8 | Makespan: 20 minutes
```

```
{0: ('R 1', [('J4', 8), ('J9', 8)]),
 2: ('R 1', [('J2', 3), ('J6', 10), ('J10', 1)]),
 1: ('R 2', [('J1', 10), ('J5', 1), ('J7', 3)]),
 4: ('R 2', [('J3', 20)]),
 3: ('R 3', [('J8', 20)])}
```

```
# Original schedule with jobs
original_schedule = rearranged_schedule


# User input for subtasks of each job
job_subtasks = {
    'J1': [('T1', 4), ('T2', 3), ('T3', 3)],
    'J2': [('T1', 3)],
    'J3': [('T1', 7), ('T2', 8), ('T3', 5)],
    'J4': [('T1', 4), ('T2', 4)],
    'J5': [('T1', 1)],
    'J6': [('T1', 5), ('T2', 5)],
    'J7': [('T1', 1), ('T2', 1), ('T3', 1)],
    'J8': [('T1', 10), ('T2', 10)],
    'J9': [('T1', 4), ('T2', 4)],
```

```
    'J10': [('T1', 1)]
}


# Replace jobs with their corresponding subtasks in the schedule
for machine, (cell, jobs) in original_schedule.items():
    new_jobs = []
    for job, _ in jobs:
        if job in job_subtasks:
            for subtask in job_subtasks[job]:
                new_subtask = (f"{subtask[0]}{job[1:]}", subtask[1])
# Format: T12 for Task 2 of Job 1
                new_jobs.append(new_subtask)
    original_schedule[machine] = (cell, new_jobs)

# Displaying the updated schedule
for machine, (cell, jobs) in original_schedule.items():
    job_list = ', '.join([f"{job[0]} ({job[1]})" for job in jobs])
    print(f"Machine {machine} (in {cell}) - Tasks: {job_list}")

# Return original_schedule for further analysis if needed
original_schedule


# Re-defining the original schedule and job subtasks due to code
execution state reset


# User input for subtasks of each job
job_subtasks = {
    'J1': [('T1', 4), ('T2', 3), ('T3', 3)],
    'J2': [('T1', 3)],
    'J3': [('T1', 7), ('T2', 8), ('T3', 5)],
    'J4': [('T1', 4), ('T2', 4)],
    'J5': [('T1', 1)],
    'J6': [('T1', 5), ('T2', 5)],
    'J7': [('T1', 1), ('T2', 1), ('T3', 1)],
    'J8': [('T1', 10), ('T2', 10)],
    'J9': [('T1', 4), ('T2', 4)],
    'J10': [('T1', 1)]
}

# Replace jobs with their corresponding subtasks in the schedule
for machine, (cell, jobs) in original_schedule.items():
    new_jobs = []
    for job, _ in jobs:
        if job in job_subtasks:
            for subtask in job_subtasks[job]:
                subtask_label = f"T{job[1:] if len(job) > 2 else
job[1]}{subtask[0][1]}"  # Correct format: TXY
```

```
                new_jobs.append((subtask_label, subtask[1]))
    original_schedule[machine] = (cell, new_jobs)

# Displaying the updated schedule
for machine, (cell, jobs) in original_schedule.items():
    job_list = ', '.join([f"{job[0]} ({job[1]})" for job in jobs])
    print(f"Machine {machine} (in {cell}) - Tasks: {job_list}")

# Return original_schedule for further analysis if needed
original_schedule
```

```
Machine 0 (in R 1) - Tasks: T81 (10), T82 (10)
Machine 1 (in R 2) - Tasks: T51 (1), T61 (5), T62 (5), T71 (1), T72
(1), T73 (1)
Machine 2 (in R 1) - Tasks: T11 (4), T12 (3), T13 (3), T21 (3), T101
(1)
Machine 3 (in R 3) - Tasks: T31 (7), T32 (8), T33 (5)
Machine 4 (in R 2) - Tasks: T41 (4), T42 (4), T91 (4), T92 (4)
```

```
{0: ('R 1', [('T81', 10), ('T82', 10)]),
 1: ('R 2',
  [('T51', 1), ('T61', 5), ('T62', 5), ('T71', 1), ('T72', 1), ('T73',
1)]),
 2: ('R 1', [('T11', 4), ('T12', 3), ('T13', 3), ('T21', 3), ('T101',
1)]),
 3: ('R 3', [('T31', 7), ('T32', 8), ('T33', 5)]),
 4: ('R 2', [('T41', 4), ('T42', 4), ('T91', 4), ('T92', 4)])}
```

```
# Re-defining the original schedule and job subtasks due to code
execution state reset

# User input for subtasks of each job
job_subtasks = {
    'J1': [('T1', 4,'S1'), ('T2', 3,'S3'), ('T3', 3),'S4'],
    'J2': [('T1', 3,'S1')],
    'J3': [('T1', 7,'S3'), ('T2', 8,'S4'), ('T3', 5,'S6')],
    'J4': [('T1', 4,'S1'), ('T2', 4,'S3')],
    'J5': [('T1', 1,'S1')],
    'J6': [('T1', 5,'S2'), ('T2', 5,'S3')],
    'J7': [('T1', 1,'S2'), ('T2', 1,'S2'), ('T3', 1,'S1')],
    'J8': [('T1', 10,'S4'), ('T2', 10,'S2')],
    'J9': [('T1', 4,'S1'), ('T2', 4,'S4')],
    'J10': [('T1', 1,'S4')]
}

# Replace jobs with their corresponding subtasks and tools in the
schedule
for machine, (cell, jobs) in original_schedule.items():
    new_jobs = []
    for job_info in jobs:
```

```python
        job, duration = job_info[0], job_info[1]  # Unpack job ID and duration
        if job in job_subtasks:
            for subtask in job_subtasks[job]:
                # Extract task number and job number more safely
                task_number = subtask[0][1:]  # Assuming task format is 'T<number>'
                job_number = job[1:]  # Extracting job number from job ID
                subtask_label = f"T{job_number}{task_number}"  # Correct format: TXY

                tool = subtask[2] if len(subtask) > 2 else 'None'  # Handling missing tool info
                new_jobs.append((subtask_label, subtask[1], tool))
    original_schedule[machine] = (cell, new_jobs)

# Displaying the updated schedule with tools
for machine, (cell, jobs) in original_schedule.items():
    job_list = ', '.join([f"{job[0]} ({job[1]} units, Tool: {job[2]})" for job in jobs])
    print(f"Machine {machine} (in {cell}) - Tasks: {job_list}")

# Return original_schedule for further analysis if needed
original_schedule
```

```
Machine 0 (in R 1) - Tasks: T41 (4 units, Tool: S1), T42 (4 units, Tool: S3), T91 (4 units, Tool: S1), T92 (4 units, Tool: S4)
Machine 1 (in R 2) - Tasks: T11 (4 units, Tool: S1), T12 (3 units, Tool: S3), T13 (3 units, Tool: None), T1 (4 units, Tool: None), T51 (1 units, Tool: S1), T71 (1 units, Tool: S2), T72 (1 units, Tool: S2), T73 (1 units, Tool: S1)
Machine 2 (in R 1) - Tasks: T21 (3 units, Tool: S1), T61 (5 units, Tool: S2), T62 (5 units, Tool: S3), T101 (1 units, Tool: S4)
Machine 3 (in R 3) - Tasks: T81 (10 units, Tool: S4), T82 (10 units, Tool: S2)
Machine 4 (in R 2) - Tasks: T31 (7 units, Tool: S3), T32 (8 units, Tool: S4), T33 (5 units, Tool: S6)
```

```
{0: ('R 1',
  [('T41', 4, 'S1'), ('T42', 4, 'S3'), ('T91', 4, 'S1'), ('T92', 4, 'S4')]),
 1: ('R 2',
  [('T11', 4, 'S1'),
   ('T12', 3, 'S3'),
   ('T13', 3, 'None'),
   ('T1', '4', 'None'),
   ('T51', 1, 'S1'),
   ('T71', 1, 'S2'),
   ('T72', 1, 'S2'),
```

```
    ('T73', 1, 'S1')]),
 2: ('R 1',
  [('T21', 3, 'S1'), ('T61', 5, 'S2'), ('T62', 5, 'S3'), ('T101', 1,
'S4')]),
 3: ('R 3', [('T81', 10, 'S4'), ('T82', 10, 'S2')]),
 4: ('R 2', [('T31', 7, 'S3'), ('T32', 8, 'S4'), ('T33', 5, 'S6')])}

# Original schedule with robotic cells and tasks


# Filter function for a specific robotic cell
def filter_schedule_by_robot_cell(schedule, cell_name):
    return {machine: (cell, tasks) for machine, (cell, tasks) in
schedule.items() if cell == cell_name}

# Example: Filtering for Robotic Cell R1
filtered_schedule_R1 =
filter_schedule_by_robot_cell(original_schedule, 'R 1')

# Displaying the filtered schedule for R1
for machine, (cell, tasks) in filtered_schedule_R1.items():
    task_list = ', '.join([f"{task[0]} ({task[1]})" for task in
tasks])
    print(f"Machine {machine} (in {cell}) - Tasks: {task_list}")

# Return filtered_schedule_R1 for further analysis if needed
filtered_schedule_R1


Machine 0 (in R 1) - Tasks: T41 (4), T42 (4), T91 (4), T92 (4)
Machine 2 (in R 1) - Tasks: T21 (3), T61 (5), T62 (5), T101 (1)

{0: ('R 1',
  [('T41', 4, 'S1'), ('T42', 4, 'S3'), ('T91', 4, 'S1'), ('T92', 4,
'S4')]),
 2: ('R 1',
  [('T21', 3, 'S1'), ('T61', 5, 'S2'), ('T62', 5, 'S3'), ('T101', 1,
'S4')])}

def calculate_makespan_and_tool_change(schedule):
    machines = {}
    makespan = 0
    tool_changeover_time = 0

    for _, (job, tasks) in schedule.items():
        current_tool = None
        current_machine = None
        machine_time = 0

        for task in tasks:
            task_id, processing_time, tool = task
```

```python
            if current_tool is None:
                current_tool = tool
                current_machine = job + current_tool
                machines[current_machine] = 0

            if current_tool != tool:
                tool_changeover_time += 5
                current_tool = tool

            if current_machine != job + current_tool:
                machine_time = machines.get(current_machine, 0)
                current_machine = job + current_tool

            machine_time += processing_time
            machines[current_machine] = machine_time
            makespan = max(makespan, machine_time)

    return makespan, tool_changeover_time

schedule = filtered_schedule_R1

makespan, tool_changeover_time =
calculate_makespan_and_tool_change(schedule)

print("Makespan:", makespan)
print("Tool Changeover Time:", tool_changeover_time)

# Remove 'R 1' from each schedule entry
new_schedule = {key: value[1] for key, value in schedule.items()}

print(new_schedule)
```

```
Makespan: 16
Tool Changeover Time: 30
{0: [('T41', 4, 'S1'), ('T42', 4, 'S3'), ('T91', 4, 'S1'), ('T92', 4,
'S4')], 2: [('T21', 3, 'S1'), ('T61', 5, 'S2'), ('T62', 5, 'S3'),
('T101', 1, 'S4')]}
```

```python
schedule = {
    0: ('R 1', [('T41', 4, 'S1'), ('T42', 4, 'S3'), ('T91', 4, 'S1'),
('T92', 4, 'S4')]),
    2: ('R 1', [('T21', 3, 'S1'), ('T61', 5, 'S2'), ('T62', 5, 'S3'),
('T101', 1, 'S4')])
}

# Remove 'R 1' from each schedule entry
new_schedule = {key: value[1] for key, value in schedule.items()}

print(new_schedule)
```