→ COGS 182 Project 2 | Schedule Optimization

▼ Checkpoint 3: Implementing the Algorithms

```
import numpy as np
import matplotlib.pyplot as plt
import sys
from mpl_toolkits import mplot3d
import seaborn as sns
from tqdm.auto import tqdm
import pickle
```

▼ List of Possible Topics & Actions (Hour Allocations to Each Topic) & Resultant States

```
possible_topics=['topology',
                 'Japanese',
                 'webtooning',
                 'physics',
                 'video-editing',
                 'graphic design',
                 'animation',
                 'psych/linguistics',
                 'R/statistical packages',
                 'front-end coding']
\#0-12 = allot hrs, 7 = commit to schedule
possible_actions = [0,1,3,5,7]
r_{times} = []
for remaining_time in np.arange(25):
    for action in [0,1,3,5]:
        for org_hrs in [0,1,2,3,4]:
            for lab_hrs in [0, 1,2]:
                r_time = remaining_time - action - org_hrs - lab_hrs
                r_times.append(r_time)
possible_remaining_times = sorted(np.unique(r_times))
terminal_states = []
for r_time in possible_remaining_times:
    terminal_state = [r_time, 0]
    terminal_states.append(terminal_state)
possible_states = [0]
for r_time in possible_remaining_times:
        for len_schedule in np.arange(len(possible_topics) +1):
            for topic in possible_topics:
                state = [r_time, len_schedule, topic]
                if state[0] > 0:
                    possible_states.append(state)
# len(terminal_states),len(possible_states)
pos_states = [0]
for state_indx, state in enumerate(possible_states[1:]):
    r_time, len_schd, topic = state
    no_topic_state = [r_time, len_schd]
    if no_topic_state not in pos_states:
        pos_states.append(no_topic_state)
```

▼ Factors (i.e. schedule fulfillment, coherence) to take into account when calculating the rewards

```
fulfillment_factors = {'topology': [1, 1, 0.6],
                       'Japanese': [0.7, 1, 1],
                        'webtooning': [0.5, 0.5, 0.4],
                        'physics': [1, 0.9, .6],
                        'video-editing': [0.8, 0.4, 1],
                        'graphic design': [0.5, 0.3, 0.6],
                        'animation': [1, 0.9, 0.7],
                        'psych/linguistics': [0.6, 0.7, 0.7],
                        'R/statistical packages': [0.5, 0.4, 1],
                        'front-end coding': [0.6, 0.7, 1]}
fulfillment_scores = {}
for topic in fulfillment_factors:
    frustration, intrigue, applicability = fulfillment_factors[topic]
    #fulfillment factors scale positive rewards
    fulfillment_score = (0.25*frustration + 0.5*intrigue + 0.25*applicability)
    fulfillment_scores[topic] = fulfillment_score
```

- ▼ Step function, take in remaining time (in a day), the schedule, the topic of consideration, and the action
 - NOTE: for feasibility sake, the observed state is actually only the LENGTH of the current schedule, but the entire schedule is passed for the environment to do calculations

```
# state keeps track of remaining free time in a day, the length of the schedule, and the current topic
    # length of schedule instead of actual schedule, since that's too many different states...
def step(remaining_time, schedule, topic, action):
    state = [remaining_time, len(schedule)]
    state_indx = pos_states.index(state)
    if action != 7:
        if action != 0:
            remaining_time -= action
            schedule[topic] = action #then allot hours according to a policy?
        topic = np.random.choice(possible_topics)
        new_state = [remaining_time, len(schedule)]
        reward = 0
        if remaining_time < 6:</pre>
            # print("\033[1;31m KAROSHI \033[0m YOU DIED OF OVERWORK")
            # sys.stdout.flush()
            reward = -1
            new_state = 0
        return new_state, reward, schedule, topic
    # ENVIRONMENT GOES when commit to schedule
    elif action == 7:
        new_state, reward, schedule, remaining_time = env(remaining_time, schedule)
        return new_state, reward, schedule, remaining_time
```

When the environment goes,

it takes the remaining time and schedule, adds on random hours from existing commitments, calculates if the remaining time in a day allows for sleep, calculates the schedule fulfillment and modulates the fulfillment with bonuses for schedule coherence and having enough to sleep. The terminal state is returned [remaining_time, 0].

```
else:
            total_fulfillment += fulfillment_scores[topic]
    total fulfillment = np.mean(total fulfillment)
    # factor in related subjects into fulfillment score (mutually reinforcing)
    # if too many subjects at once, coherence turns into distraction factor, negative rate
    coherence = 1
    if len(schedule) <= 3:</pre>
        for topic in schedule:
            for possible_topic in possible_topics:
                if topic == possible_topic:
                    for related_topic in related_topics[topic]:
                        if related_topic in schedule:
                            coherence += 0.05
    elif len(schedule) > 4: #too many topics
        coherence = -0.5
    # sleep and free time bonus (a rate)
    if remaining_time >= 8 and remaining_time <= 10:</pre>
        bonus = 1.1
                           # 10% bonus fulfillment
    else:
        bonus = 1
                             # no bonus
    # reward to return as a function of bonus, topic-based fulfillment and coherence of schedule
    reward = (total_fulfillment)* ((bonus + coherence)/ 2)
schedule['Orgs'] = existing_commitments['Orgs']
schedule['LabStuff'] = existing_commitments['LabStuff']
new_state = 0
return new_state, reward, schedule, remaining_time
```

▼ Algorithm 1: On-Policy First-Visit Monte Carlo Control

```
def MonteCarlo(init, num_runs):
    # num of times that action a has been selected from state s
    N = np.zeros((len(pos_states), len(possible_actions)))
   N_0 = 1
    # hyperparameters
    gamma = 0.5  # discount factor - balance between being too myopic (0) and too farsighted (1)
    epsilon = N_0/(N_0 + 0)
    # initialize policy, q_values, returns
    q_values = init_q_values(init)
    returns = np.zeros((len(pos_states), len(possible_actions)))
    policy = np.ones((len(pos_states), len(possible_actions))) * (epsilon/len(possible_actions))
    all_episodes = []
    all_episode_rewards = []
    for run in tqdm((range(num_runs)), position=0):
        remaining_time = 24
        schedule = {}
        topic = np.random.choice(possible_topics)
                   = [remaining_time, len(schedule)]
        state_indx = pos_states.index(state)
        action = np.random.choice(possible_actions, p= policy[state_indx])
        episode = []
        episode_rewards = []
        while True:
            if action != 7:
```

```
new_state, reward, schedule, topic = step(remaining_time, schedule, topic, action)
            elif action == 7:
                new_state, reward, schedule, remaining_time = step(remaining_time, schedule, topic, action)
            ep = [state, action, reward, schedule]
            episode.append(ep)
            episode_rewards.append(reward)
            if new_state == 0:
                break
            remaining_time, len_schedule = new_state
            state = [remaining_time, len_schedule]
            state_indx = pos_states.index(state)
            action = np.random.choice(possible_actions, p= policy[state_indx])
            action_indx = possible_actions.index(action)
        all_episodes.append(episode)
        all_episode_rewards.append(episode_rewards)
        G = 0
        #loop through episode in reverse
        for indx_rev, state_action in enumerate(episode[::-1]):
            state, action, reward, schedule = state_action
            state_indx = pos_states.index(state)
            action_indx = possible_actions.index(action)
            indx_state_action = len(episode_rewards) - indx_rev -1
            G += gamma*G + episode_rewards[indx_state_action]
            returns[state_indx][action_indx] = G
            if state_action not in episode[:indx_state_action]:
                N[state_indx][action_indx] += 1
                alpha = 1/(N[state_indx][action_indx]) # use time-varying alpha
                q_values[state_indx][action_indx] += alpha * (G - q_values[state_indx][action_indx])
                {\tt epsilon = N_0/(N_0 + np.min(N[state\_indx][action\_indx])) \  \  \, \# \  \, epsilon-greedy \  \, exploration \  \, strategy}
                #update policy
                optimal_action = np.argmax(q_values[state_indx])
                for action_indx in range(len(possible_actions)):
                    if action_indx == optimal_action:
                        policy[state_indx][action_indx] = 1 - epsilon + (epsilon/ len(possible_actions))
                    elif action_indx != optimal_action:
                        policy[state_indx][action_indx] = (epsilon/ len(possible_actions))
    return policy, q_values, all_episodes, all_episode_rewards
def quick_check(q_values, policy):
    #quick check of Q-values vs. policy
    print("<<QUICK CHECKS>>")
    oops = [100, 150, 200, 250]
    for oop in oops:
        print("----")
        print("state:", pos_states[oop], "\nQs:", q_values[oop],"\nPi:", policy[oop])
        print("Q says", np.argmax(q_values[oop]), "| Pi says", np.argmax(policy[oop]))
    agreements = 0
    for state_indx, state in enumerate(pos_states):
        if np.argmax(q_values[state_indx]) == np.argmax(policy[state_indx]):
            agreements += 1
    print("Policy aligns with Q-values:", agreements / len(pos_states) * 100, "%")
def get_opt_actions(policy):
```

```
def get_opt_actions(policy):

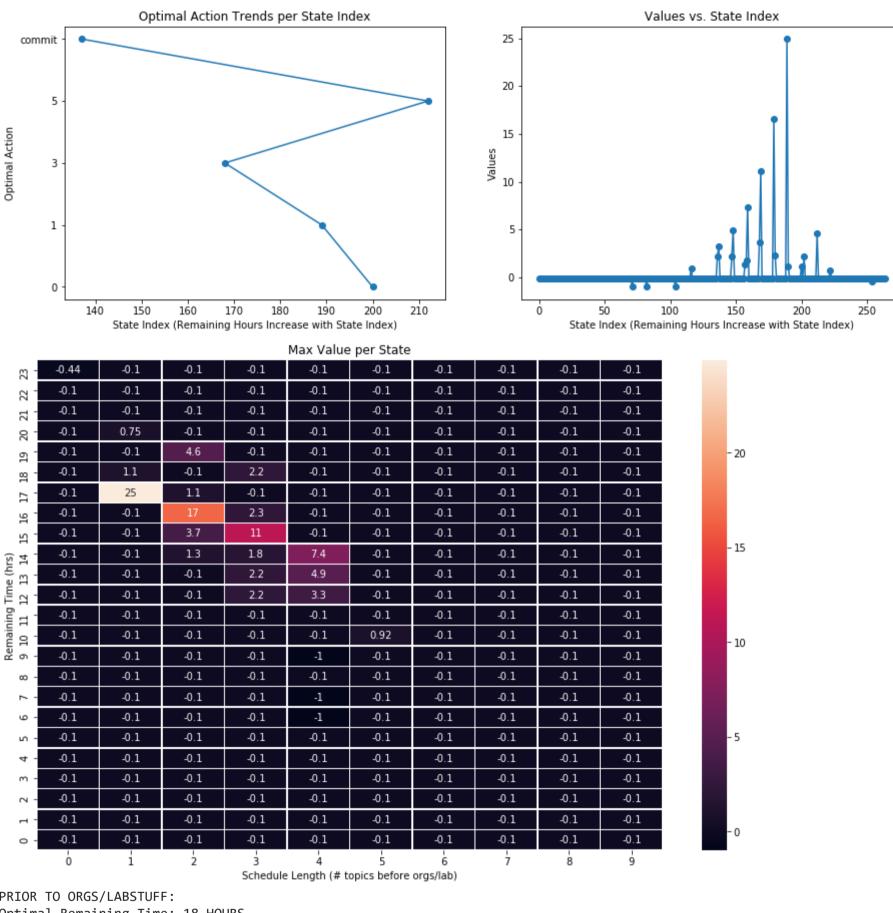
    if algo == "MC":
        optimal_actions = []
        for state_indx, state in enumerate(pos_states):
            action_indx = np.argmax(policy[state_indx])
        if action_indx == 0:
            action = 0
            optimal_actions.append(action)
        elif action_indx == 1:
            action = 1
            optimal_actions.append(action)
        elif action_indx == 2:
            action = 3
            optimal_actions.append(action)
        elif action_indx == 3:
            action = 5
```

```
optimal_actions.append(action)
            elif action indx == 4:
                action = 7
                optimal_actions.append(action)
    return optimal_actions
def calc_V_star(q_values):
   #calculate optimal value function V_star
    V_star = []
    for state_indx in range(len(pos_states)):
        optimal_value = np.max(q_values[state_indx])
        V_star.append(optimal_value)
    len(V_star)
    V_star = np.array(V_star)
    return V_star
def plot_values_actions_states(algo, values_or_qvalues):
    if algo == "MC":
        V_star = calc_V_star(values_or_qvalues)
        fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(15,5))
        ax[0].plot(np.argmax(values_or_qvalues, axis=0),["0","1","3","5","commit"], "-o")
        ax[0].set_xlabel("State Index (Remaining Hours Increase with State Index)")
        ax[0].set_ylabel("Optimal Action")
        ax[0].set_title("Optimal Action Trends per State Index")
        ax[1].plot(V_star, '-o')
        ax[1].set_ylabel("Values")
        ax[1].set_xlabel("State Index (Remaining Hours Increase with State Index)")
        ax[1].set_title("Values vs. State Index")
    elif algo == "TD":
        plt.plot(values_or_qvalues, '-o')
        plt.ylabel("Values")
        plt.xlabel("State Index (Remaining Hours Increase with State Index)")
        plt.title("Values vs. State Index")
    plt.show()
def plot_heatmap(V_star):
    x = np.linspace(0, 24, 25) #remaining times
   y = np.linspace(0, 10, len(possible_topics)) #possible schedule lengths
   X, Y = np.meshgrid(x, y)
    Z = np.ones((25, len(possible_topics)))
    for state_indx, state in enumerate(pos_states[1:]):
        r_time, len_schd = state
        value = V_star[state_indx]
        Z[r\_time - 1][len\_schd - 1] = value
    fig = plt.figure(figsize =(14, 9))
    ax = sns.heatmap(Z, linewidth=0.5, annot=True)
    plt.xlabel("Schedule Length (# topics before orgs/lab)")
    plt.ylabel("Remaining Time (hrs)")
    plt.title("Max Value per State")
    plt.xlim(0,)
    plt.ylim(0,24)
    plt.show()
    # x.shape, y.shape, Z.shape
def get_opt_schedules(V_star, all_episodes, optimal_actions):
    optimal_schedules = []
    part_optimal_schedules = []
    optimal_schd_rewards = []
    part_opt_schd_rewards = []
    for state_indx, action in enumerate(optimal_actions):
        if action == 7:
            for episode in all_episodes:
                state, action, reward, schedule = episode[-1]
                if len(schedule) > 2 and "LabStuff" in schedule:
                    if state == pos_states[np.argmax(V_star)]:
                        optimal_schedules.append(schedule)
                        optimal_schd_rewards.append(reward)
                    elif state[0] == pos_states[np.argmax(V_star)][0] or state[1] == pos_states[np.argmax(V_star)][1]:
                        part_optimal_schedules.append(schedule)
                        part_opt_schd_rewards.append(reward)
```

▼ Run Monte Carlo using different Initializations

```
%%time
num_runs = 100000
#try with different initialization of q_values
inits = [-0.1, 0.1, 'zeros', 'arb']
policies = []
MC_init_qs = []
MC_init_eps = []
MC_init_ep_rewards = []
MC_init_opt_actions = []
for init in inits:
    print("\033[1;31mINIT Q-VALUES AS {}: \033[0m\n".format(init))
    policy, q_values, all_episodes, all_episode_rewards = MonteCarlo(init, num_runs)
    optimal_actions = get_opt_actions(policy)
    policies.append(policies)
    MC_init_qs.append(q_values)
    MC_init_eps.append(all_episodes)
    MC_init_ep_rewards.append(all_episode_rewards)
    MC_init_opt_actions.append(optimal_actions)
    print()
      quick_check(q_values,policy)
      for action in possible_actions:
          if action != 7:
             print("optimal to allot",action, "hrs:", optimal_actions.count(action), "x")
             print("Optimal to commit:
                                           ", optimal_actions.count(action), "x")
      print()
    #calculate V_star & Plots
    V_star = calc_V_star(q_values)
    plot_values_actions_states("MC", q_values)
    plot_heatmap(V_star)
    best_r_time, best_len = pos_states[np.argmax(V_star)]
    print("PRIOR TO ORGS/LABSTUFF:")
    print("Optimal Remaining Time: {} HOURS \nOptimal Schedule Length: {} COMMITMENTS".format(best_r_time, best_len))
    print()
    # returns schedules that are EITHER the optimal length or optimal, get unique entries
    optimal_schedules, part_optimal_schds, opt_schd_rewards, part_opt_rewards = get_opt_schedules(V_star, all_episodes, optimal_actions)
    optimal_schedules = [dict(entry) for entry in set(frozenset(schd.items()) for schd in optimal_schedules)]
    part_optimal_schds = [dict(entry) for entry in set(frozenset(schd.items()) for schd in part_optimal_schds)]
    print("WITH ORGS/LABS IN FINAL CONSIDERATION")
    print("Optimal Schedules:")
    for indx, schd in enumerate(optimal_schedules):
        print(schd, "| Fulfillment:", opt_schd_rewards[indx])
    if optimal_schedules == []:
        print("Partially Optimal Schedules:")
        for indx, schd in enumerate(part_optimal_schds):
            print(schd, "| Fulfillment:", part_opt_rewards[indx])
    print("-----\n")
    sys.stdout.flush()
```

HBox(children=(IntProgress(value=0, max=100000), HTML(value='')))



PRIOR TO ORGS/LABSTUFF:

Optimal Remaining Time: 18 HOURS Optimal Schedule Length: 1 COMMITMENTS

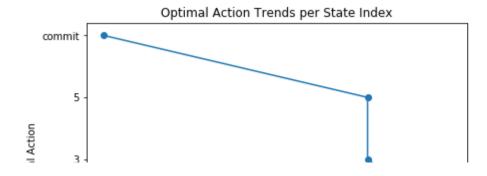
WITH ORGS/LABS IN FINAL CONSIDERATION

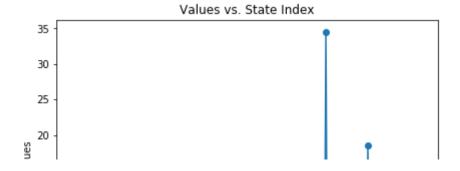
Optimal Schedules:

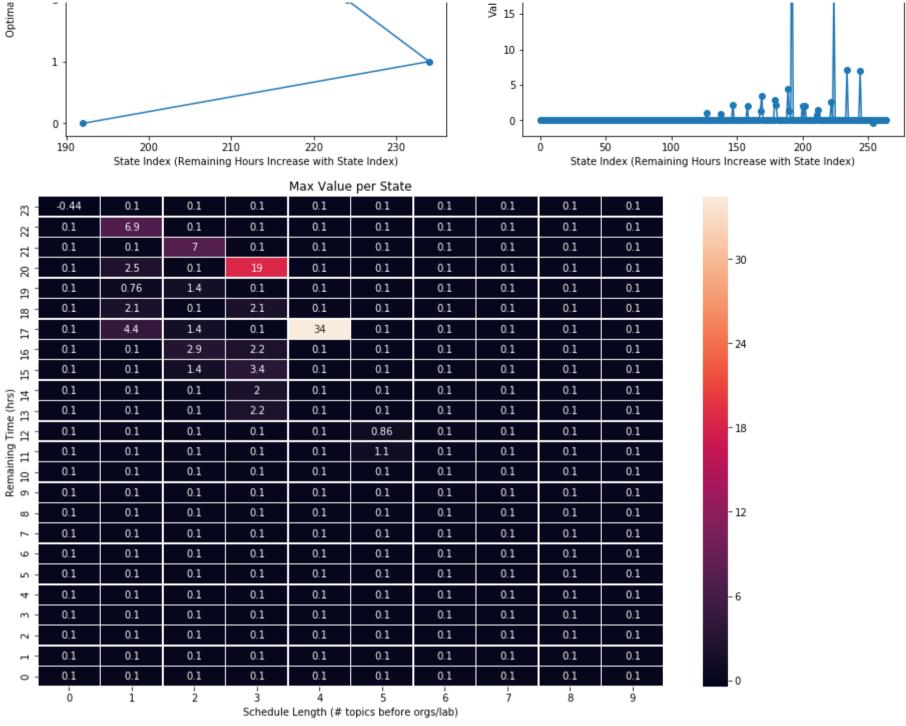
Partially Optimal Schedules:

{'Orgs': 3, 'webtooning': 3, 'LabStuff': 1} | Fulfillment: 0.9 {'Orgs': 4, 'topology': 5, 'LabStuff': 1} | Fulfillment: 1.15 {'Orgs': 1, 'topology': 5, 'LabStuff': 1} | Fulfillment: 0.875 {'Orgs': 4, 'animation': 3, 'LabStuff': 2} | Fulfillment: 0.9 {'Orgs': 2, 'webtooning': 1, 'psych/linguistics': 5, 'LabStuff': 2} | Fulfillment: 0.9 {'animation': 5, 'Orgs': 1, 'LabStuff': 2} | Fulfillment: 0.475 {'Orgs': 4, 'R/statistical packages': 5, 'LabStuff': 1} | Fulfillment: 0.875 {'Orgs': 2, 'LabStuff': 2, 'topology': 3} | Fulfillment: 0.575

INIT Q-VALUES AS 0.1:







PRIOR TO ORGS/LABSTUFF:

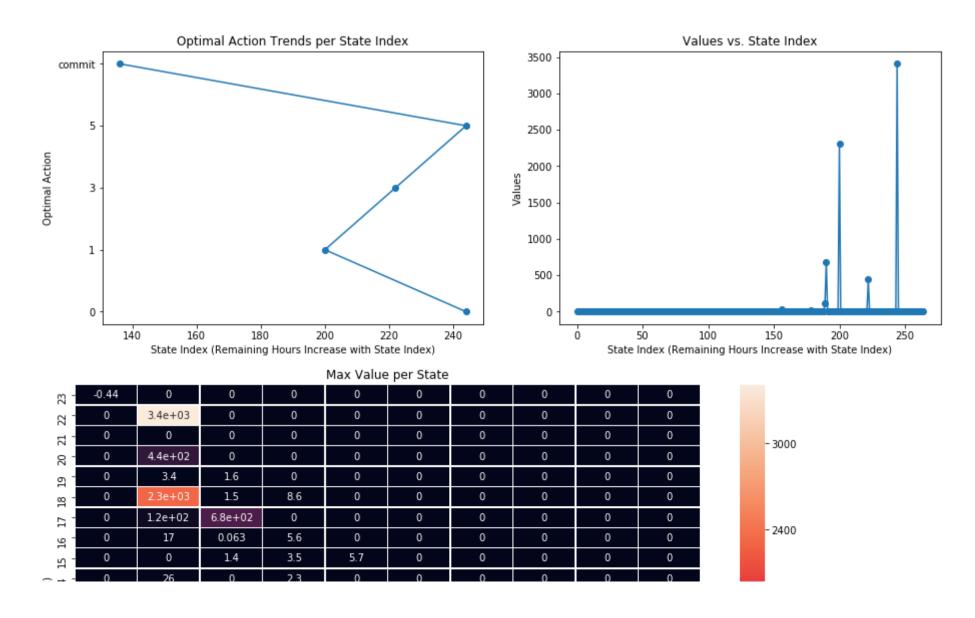
Optimal Remaining Time: 18 HOURS
Optimal Schedule Length: 4 COMMITMENTS

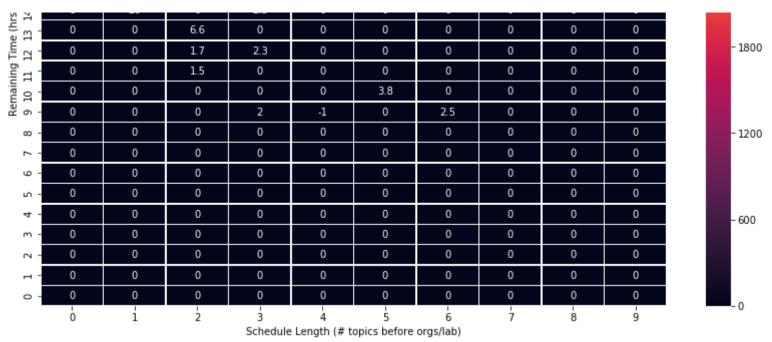
WITH ORGS/LABS IN FINAL CONSIDERATION

Optimal Schedules:

('Japanese': 3, 'LabStuff': 1, 'topology': 1, 'Orgs': 3, 'R/statistical packages': 1, 'physics': 1} | Fulfillment: 3.25

INIT Q-VALUES AS zeros:





PRIOR TO ORGS/LABSTUFF:

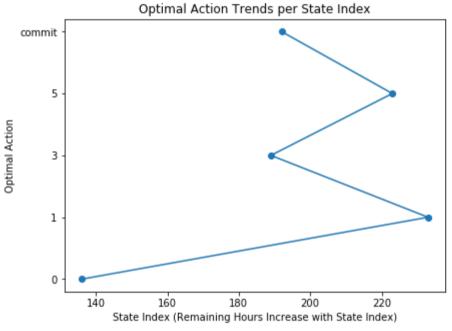
Optimal Remaining Time: 23 HOURS
Optimal Schedule Length: 1 COMMITMENTS

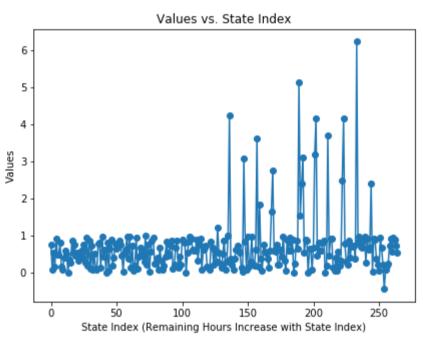
WITH ORGS/LABS IN FINAL CONSIDERATION

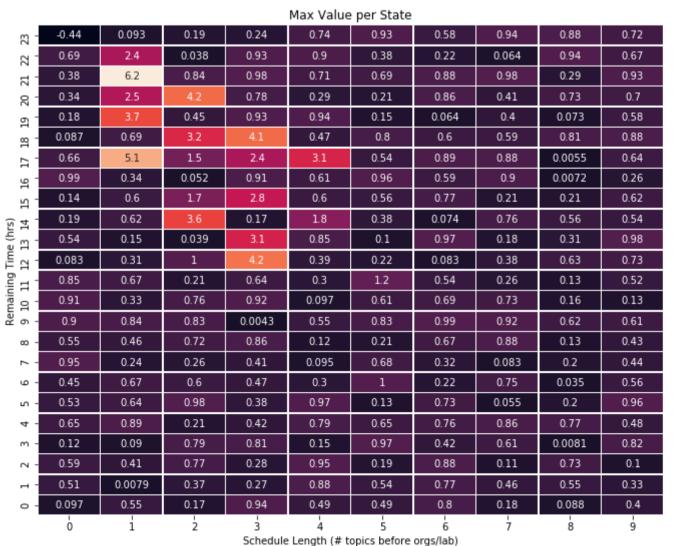
Optimal Schedules:

{'LabStuff': 1, 'Orgs': 2, 'Japanese': 1} | Fulfillment: -0.5 {'topology': 1, 'Orgs': 2, 'LabStuff': 2} | Fulfillment: -0.5

INIT Q-VALUES AS arb:







- 6.0
- 4.5
- 3.0
- 1.5
- 0.0

```
with open('MC_init_qs.pickle', 'wb') as mc_results2:
    pickle.dump(MC_init_qs, mc_results2)
with open('MC_init_eps.pickle', 'wb') as mc_results3:
    pickle.dump(MC_init_eps, mc_results3)
with open('MC_init_ep_rewards.pickle', 'wb') as mc_results4:
    pickle.dump(MC_init_ep_rewards,mc_results4)
with open('MC_init_opt_actions.pickle', 'wb') as mc_results5:
    pickle.dump(MC_init_opt_actions, mc_results5)
```

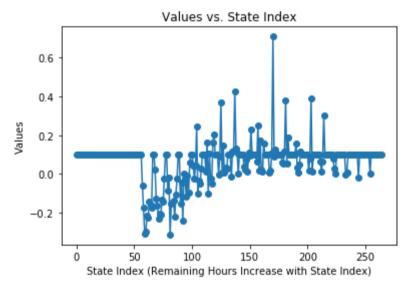
- Algorithm 2: $TD(\lambda)$

```
# epsilon-greedy policy
def epsilon_greedy(epsilon, values, state):
    state_indx = pos_states.index(state)
    be_greedy = (np.random.random() > epsilon)
    if be_greedy:
        action = np.argmax(values[state_indx]) #optimal action
    else:
        action = np.random.choice(possible_actions)
    return action
# # initialize V(s)
def init_values(init):
    if init == "zeros":
        values = np.zeros((len(pos_states)))
    elif init == 'arb':
        values = np.ones((len(pos_states)))
        for state_indx,state in enumerate(pos_states):
            if state in terminal_states:
                values[state_indx] = 0
            else:
                values[state_indx] = np.random.random()
    else:
        values = np.ones((len(pos_states))) * init
    return values
def TD(init, lmbda, num_eps):
    \# initialize V(s) arbitrarily but set to 0 if state is terminal
    values = init_values(init)
    all_episodes = []
    all_episode_rewards = []
    for episode in tqdm(range(num_eps)):
        #initialize weights
        e_weights = np.zeros(len(pos_states))
        #initialize S
        remaining_time = 24
        schedule = {}
        topic = np.random.choice(possible_topics)
                   = [remaining_time, len(schedule)]
        state_indx = pos_states.index(state)
        episode = []
        episode_rewards = []
        # for each step in episode
        while True:
            #take action, observe reward, new_state
            action = epsilon_greedy(epsilon, values, state)
            action_indx = possible_actions.index(action)
            if action == 7:
                new_state, reward, schedule, remaining_time = step(remaining_time, schedule, topic, action)
                new_state_indx = pos_states.index(new_state)
            elif action != 7:
                new_state, reward, schedule, topic = step(remaining_time, schedule, topic, action)
                new_state_indx = pos_states.index(new_state)
            ep = [state, action, reward, schedule]
            episode.append(ep)
            episode_rewards.append(reward)
            # update error and weights
                                 = reward + gamma*values[new_state_indx] - values[state_indx]
            e_weights[state_indx] = (1 - alpha) * e_weights[state_indx] + 1 #dutch traces
            #update values and eligibility weights for all states
                                 = values + alpha * td error * e weights
            values
                              = gamma * lmhda * e weights
```

```
%%time
lmbdas = [0,0.5,0.75,1]
epsilon = 0.1
alpha = 0.1
init = 'arb'
inits = [0.1, 'arb', 'zeros']
num_eps = 100000 # number of runs/eps
TD_init_values = []
TD_init_eps = []
TD_init_ep_rewards = []
TD_init_opt_actions = []
for init in inits:
    print("\033[1;38m INIT VALUES as {} \033[0m".format(init))
    for lmbda in lmbdas:
        print("\033[1;43m LAMBDA = {} \033[0m".format(lmbda))
        values, all_episodes, all_episode_rewards = TD(init, lmbda, num_eps)
        optimal_actions = get_opt_actions("TD", values)
        TD_init_values.append(values)
        TD_init_eps.append(all_episodes)
        TD_init_ep_rewards.append(all_episode_rewards)
        TD_init_opt_actions.append(optimal_actions)
        #plot stuff
        plot_heatmap(values)
        plot_values_actions_states("TD", values)
        if np.argmax(values) != 0:
           best_r_time, best_len = pos_states[np.argmax(values)]
            print("PRIOR TO ORGS/LAB:")
            print("Optimal Free Time: {} HOURS \nOptimal Schedule Length (without Orgs/Lab): {} COMMITMENTS".format(best_r_time, best_len))
        # returns schedules that are EITHER the optimal length or optimal, get unique entries
        optimal_schedules, part_optimal_schds, opt_schd_rewards, part_opt_rewards = get_opt_schedules(values, all_episodes, optimal_actions
        optimal_schedules = [dict(entry) for entry in set(frozenset(schd.items()) for schd in optimal_schedules)]
        part_optimal_schds = [dict(entry) for entry in set(frozenset(schd.items()) for schd in part_optimal_schds)]
        print("FACTORING IN ORGS/LAB:")
        print("Optimal Schedules:")
        for indx, schd in enumerate(optimal_schedules):
            print(schd, "| Fulfillment:", opt_schd_rewards[indx])
        print()
        if optimal_schedules == []:
           print("Partially Optimal Schedules:")
            for indx, schd in enumerate(part_optimal_schds):
                print(schd, "| Fulfillment:", part_opt_rewards[indx])
        print("-----\n")
        sys.stdout.flush()
```

HBox(children=(IntProgress(value=0, max=100000), HTML(value='')))

	1	1	1	1	Max V alue	per State	1	1	1	1	
23	0.0044	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	- 1.00
22	0.1	-0.017	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
77	0.1	-0.0028	0.0094	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
20 -	0.1	0.079	0.031	0.00057	0.1	0.1	0.1	0.1	0.1	0.1	
13	0.1	0.066	0.014	0.065	0.3	0.1	0.1	0.1	0.1	0.1	- 0.75
81	0.1	0.098	0.019	0.079	0.39	0.011	0.1	0.1	0.1	0.1	
17	0.1	0.099	0.16	0.035	0.0073	0.05	0.12	0.1	0.1	0.1	
16	0.1	0.061	0.055	0.12	0.38	0.054	0.19	0.1	0.1	0.1	
15	0.1	0.0057	0.018	0.11	0.71	0.09	0.13	0.1	0.1	0.1	
	0.1	0.066	0.25	0.017	0.17	0.024	0.01	0.16	0.1	0.1	- 0.50
e (hi	0.1	0.039	0.022	0.072	0.015	0.11	0.095	0.23	0.1	0.1	
II 7	0.1	-0.014	0.12	0.12	0.43	0.13	0.11	0.002	0.1	0.1	
Remaining 7 9 10 11 1	0.1	-0.00057	0.088	0.37	0.0052	0.15	0.0059	0.035	0.026	0.1	
mair 10	0.1	0.013	0.16	-0.1	0.11	-0.023	-0.051	0.16	0.2	0.1	- 0.25
9 -	0.1	0.045	-0.023	0.043	0.25	-0.1	-0.033	-0.047	0.029	0.1	
ω -	0.1	-0.15	-0.044	-0.24	0.00047	-0.023	-0.12	-0.0095	-0.095	0.058	
7	0.1	-0.087	-0.017	-0.31	-0.15	-0.15	-0.14	-0.22	-0.1	-0.025	
9 -	0.1	0.024	-0.13	-0.16	-0.23	-0.21	-0.21	-0.13	-0.14	-0.022	
- 22	0.1	-0.059	-0.17	-0.31	-0.3	-0.21	-0.22	-0.14	-0.16	-0.17	- 0.00
4 -	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
m -	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
2 -	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.25
0 -	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.23
	ó	í	2	3	4	5	6	7	8	9	
				Schedule	Length (# t	opics before	orgs/lab)				



PRIOR TO ORGS/LAB:

Optimal Free Time: 16 HOURS

Optimal Schedule Length (without Orgs/Lab): 4 COMMITMENTS

FACTORING IN ORGS/LAB: Optimal Schedules:

Partially Optimal Schedules:

LAMBDA = 0.5
HBox(children=(IntProgress(value=0, max=100000), HTML(value='')))

	1	1	1	1	Max V alue	per S tate	1	1	1	1
g -	-0.058	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
22 -	0.1	0.00063	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
71	0.1	-0.001	0.27	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2 -	0.1	0.011	-0.016	0.22	0.1	0.1	0.1	0.1	0.1	0.1
е Б	0.1	0.023	0.029	0.33	0.07	0.1	0.1	0.1	0.1	0.1
殿 -	0.1	0.096	0.33	0.014	0.13	0.0075	0.1	0.1	0.1	0.1
17	0.1	0.084	0.2	0.29	0.36	0.024	0.0098	0.1	0.1	0.1
16	0.1	0.082	0.15	0.0083	0.52	0.044	0.21	0.015	0.1	0.1
55 -	0.1	0.095	0.0061	0.0087	0.018	0.064	0.0013	0.1	0.1	0.1
	0.1	0.094	0.19	-0.0041	0.37	0.025	0.0095	0.18	0.1	0.1
Time (hrs) 12 13 14	0.1	0.0021	0.011	0.21	0.0079	0.0025	0.015	0.038	0.1	0.1
트겁	0.1	-0.0007	0.24	0.5	0.0033	0.083	0.18	0.0048	0.1	0.1
ing 11	0.1	0.11	0.18	0.005	0.27	0.13	-0.039	0.19	0.1	0.1
Remaining 9 10 11	0.1	-0.047	0.0022	0.25	-0.0074	-0.019	0.098	0.27	0.1	0.1
- 9 Re	0.1	-0.028	-0.13	-0.044	-0.028	0.068	-0.21	-0.095	0.26	0.1
∞ -	0.1	-0.03	-0.031	-0.093	-0.0058	-0.26	-0.21	-0.019	0.098	-0.028
۲-	0.1	-0.12	-0.16	-0.24	-0.16	-0.23	-0.18	-0.13	-0.0079	0.053
9 -	0.1	0.1	-0.23	-0.37	-0.18	-0.32	-0.24	-0.35	-0.2	-0.18
ı. د	0.1	-0.2	-0.28	-0.16	-0.23	-0.18	-0.19	-0.36	-0.16	-0.054

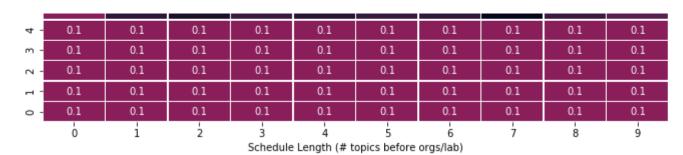
- 1.00

- 0.75

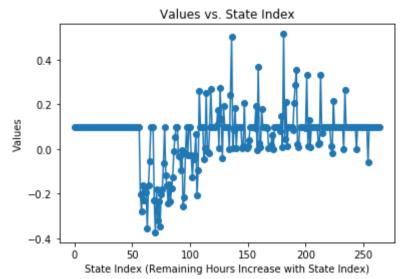
- 0.50

- 0.25

- 0.00







Optimal Free Time: 17 HOURS

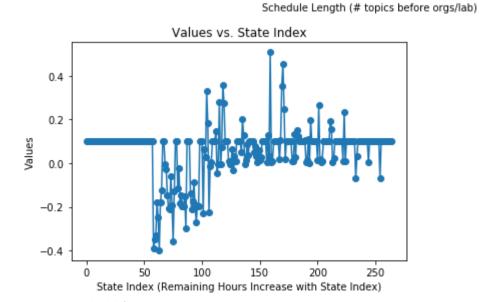
Optimal Schedule Length (without Orgs/Lab): 4 COMMITMENTS

FACTORING IN ORGS/LAB: Optimal Schedules:

Partially Optimal Schedules:

LAMBDA = 0.75
HBox(children=(IntProgress(value=0, max=100000), HTML(value='')))

	1	1	1	1	Max V alue	per S tate	1	1	1	1	1.00
23	-0.07	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	- 1.00
72	0.1	0.0058	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
72	0.1	-0.068	0.031	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
22	0.1	0.0091	0.23	0.01	0.1	0.1	0.1	0.1	0.1	0.1	
139	0.1	0.19	0.16	0.0048	0.024	0.1	0.1	0.1	0.1	0.1	- 0.75
8 -	0.1	0.013	0.26	0.017	0.0065	0.011	0.1	0.1	0.1	0.1	
17	0.1	0.11	0.0069	0.012	0.11	0.00082	0.2	0.1	0.1	0.1	
16	0.1	0.0095	0.01	0.13	0.028	0.15	0.12	0.1	0.1	0.1	
15	0.1	0.016	0.11	0.36	0.46	0.25	0.018	0.1	0.1	0.1	- 0.50
	0.1	0.006	0.13	0.032	0.51	0.0042	0.011	0.1	0.1	0.1	
e (hr 13	0.1	0.05	0.078	0.034	0.0035	0.062	0.013	0.028	0.1	0.1	
Tim 12	0.1	0.049	0.2	0.13	-0.0041	0.015	0.089	0.036	0.1	0.1	
ing 11	0.1	0.0085	-0.0027	-0.0032	0.063	-0.03	0.031	0.01	0.1	0.1	- 0.25
Remaining 9 10 11	0.1	0.15	-0.045	-0.0044	0.28	-0.0061	0.073	0.36	0.28	0.1	0.23
Rer 9	0.1	-0.23	0.065	0.028	0.33	0.18	-0.23	-0.016	0.0063	0.1	
∞ -	0.1	-0.14	-0.21	-0.17	-0.089	-0.2	-0.27	-0.19	-0.2	0.1	
7	0.1	-0.11	-0.024	-0.18	-0.15	-0.2	-0.2	-0.15	-0.3	0.1	
9 -	0.1	-0.005	-0.029	-0.14	-0.21	-0.21	-0.059	-0.19	-0.36	-0.13	- 0.00
٠ کا	0.1	0.1	-0.39	-0.35	-0.33	-0.18	-0.25	-0.4	-0.18	-0.12	
4 -	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
m -	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
2 -	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.25
П -	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
0 -	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
,	ó	í	2	3	4	5	6	7	8	9	
				Cabadula	Langth /# t	onice before	acae/lab\				

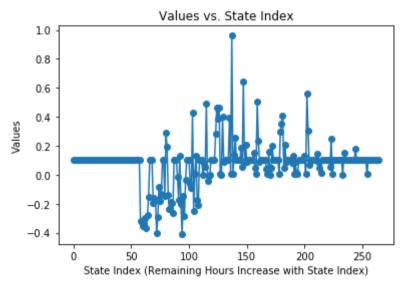


PRIOR TO ORGS/LAB:
Optimal Free Time: 15 HOURS

Optimal Schedule Length (without Orgs/Lab): 4 COMMITMENTS

LAMBDA = 1
HBox(children=(IntProgress(value=0, max=100000), HTML(value='')))

	1	1	1	1	Max V alue	per State	1	1	1	1	1.00
23	0.0048	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	- 1.00
72	0.1	0.18	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
77	0.1	0.00067	0.15	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
20	0.1	0.052	0.25	0.006	0.1	0.1	0.1	0.1	0.1	0.1	
19	0.1	0.14	0.087	0.047	0.013	0.1	0.1	0.1	0.1	0.1	- 0.75
8 -	0.1	0.13	0.0068	0.56	0.31	0.065	0.1	0.1	0.1	0.1	
17	0.1	0.084	0.13	0.0069	0.086	0.01	0.0057	0.1	0.1	0.1	
16	0.1	0.0043	0.3	0.36	0.41	0.049	0.21	0.09	0.1	0.1	
15	0.1	0.042	0.0027	0.16	0.00065	0.049	0.2	0.1	0.1	0.1	- 0.50
	0.1	0.15	0.046	0.0059	0.51	0.24	0.092	0.1	0.1	0.1	
Time (hrs)	0.1	0.18	0.056	0.64	0.084	0.21	0.086	0.1	0.1	0.1	
Tim 12	0.1	0.11	0.39	0.0043	0.96	0.0072	0.13	0.25	0.1	0.1	
Remaining 7 9 10 11 1	0.1	0.28	0.46	0.39	0.47	0.0063	-0.0011	0.4	0.09	0.1	- 0.25
mair 10	0.1	0.00084	0.079	0.058	0.49	-0.044	-0.0079	-0.0032	0.1	0.1	
8 e -	0.1	-0.061	-0.088	0.43	-0.25	0.0055	0.13	-0.17	-0.21	0.1	
∞ -	0.1	-0.017	-0.18	0.13	-0.2	-0.41	-0.14	-0.29	-0.038	0.1	
7	0.1	-0.15	0.29	0.19	-0.14	-0.24	-0.19	-0.2	-0.27	0.1	- 0.00
9 -	0.1	0.1	-0.2	-0.16	-0.17	-0.4	-0.29	-0.085	-0.18	-0.14	- 0.00
20 -	0.1	0.1	-0.32	-0.33	-0.36	-0.34	-0.3	-0.37	-0.28	-0.15	
4 -	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
m -	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
2	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.25
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
0 -	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
	ó	i	2	3	4	5	6	7	8	9	_ _
				Schedule	Length (# to	opics before	orgs/lab)				



Optimal Free Time: 13 HOURS

Optimal Schedule Length (without Orgs/Lab): 4 COMMITMENTS

FACTORING IN ORGS/LAB: Optimal Schedules:

Partially Optimal Schedules:

INIT VALUES as arb LAMBDA = 0

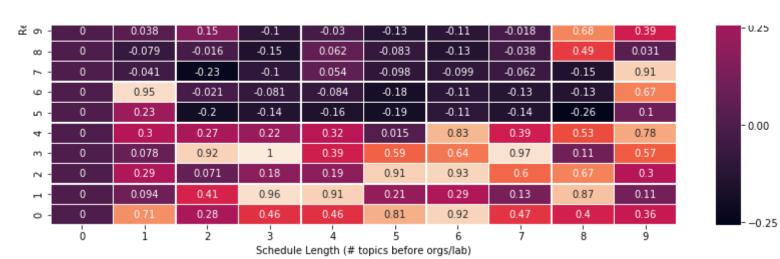
HBox(children=(IntProgress(value=0, max=100000), HTML(value='')))

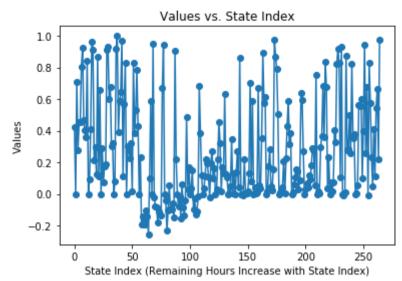
	1	1	1	1	Max V alue	per S tate	1	1	1	1
ღ -	-0.011	0.83	0.58	0.23	0.051	0.41	0.11	0.55	0.66	0.22
22 -	0	0.0021	0.052	0.56	0.55	0.6	0.098	0.4	0.95	0.26
77	0	-0.0099	0.012	0.87	0.28	0.28	0.5	0.26	0.82	0.35
g -	0	0.073	0.018	0.072	0.4	0.33	0.83	0.92	0.83	0.11
ള -	0	0.017	0.0054	0.3	0.36	0.68	0.36	0.84	0.68	0.038
g -	0	0.03	0.069	0.042	0.12	0.091	0.18	0.29	0.27	0.057
71	0	0.012	0.059	0.1	0.16	0.15	0.031	0.089	0.64	0.59
91 -	0	0.0082	0.02	0.042	0.21	0.0067	0.22	0.43	0.59	0.31
<u>12</u>	0	0.016	0.17	0.28	0.049	0.023	0.16	0.98	0.87	0.79
47	0	0.042	0.00047	0.01	0.67	0.057	0.036	0.35	0.89	0.58
<u>т</u>	0	-0.0032	-0.0061	0.22	0.001	0.00088	0.014	0.13	0.7	0.082
27	0	0.0044	0.049	0.35	0.0037	0.13	-0.00026	0.14	0.27	0.039
<u> </u>	0	0.03	0.22	0.45	0.32	0.019	0.17	0.13	0.63	0.13
요 -	0	0.033	0.11	0.22	0.2	0.1	-0.02	0.2	0.27	0.1

- 1.00

- 0.75

- 0.50





Optimal Free Time: 4 HOURS

Optimal Schedule Length (without Orgs/Lab): 3 COMMITMENTS

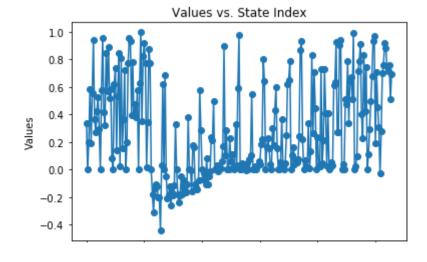
FACTORING IN ORGS/LAB:
Optimal Schedules:

Partially Optimal Schedules:

LAMBDA = 0.5
HBox(children=(IntProgress(value=0, max=100000), HTML(value='')))

	1	1	1	1	Max V alue	per S tate	1	1	1	1	1.00
23	-0.029	0.28	0.7	0.76	0.92	0.89	0.75	0.72	0.76	0.51	- 1.00
72	0	0.11	0.3	0.5	0.68	0.94	0.97	0.19	0.71	0.45	
21	0	0.024	0.098	0.72	0.79	0.91	0.4	0.23	0.83	0.43	
20	0	0.042	0.0052	0.51	0.47	0.79	0.34	0.65	0.67	0.51	
13	0	0.02	0.05	0.033	0.61	0.64	0.92	0.27	0.27	0.9	- 0.75
18	0	0.24	0.031	0.021	0.73	0.046	0.21	0.73	0.41	0.023	
17	0	0.066	0.018	0.017	0.34	0.0072	0.13	0.83	0.26	0.71	
16	0	0.11	0.16	0.038	0.077	0.056	0.078	0.25	0.87	0.93	
15	0	0.0019	0.055	0.37	0.016	0.0045	0.05	0.25	0.62	0.65	- 0.50
	0	0.17	0.23	0.15	0.039	0.0022	0.3	0.18	0.44	0.6	
Time (hrs) 12 13 14	0	0.013	0.027	0.0045	0.021	0.058	0.038	0.19	0.22	0.8	
Tim(0	0.0053	0.045	0.026	0.029	-0.0062	0.045	-0.00079	0.077	0.11	
ing 11	0	0.0023	0.23	0.062	0.11	0.018	0.0013	0.33	0.03	0.6	- 0.25
Remaining 9 10 11 1	0	0.01	0.032	-0.013	0.004	0.0078	0.063	0.17	0.9	0.11	
Rer 9	0	-0.07	-0.024	-0.11	0.086	-0.11	-0.052	-0.0095	0.24	0.21	
ω -	0	-0.0034	-0.15	0.18	0.16	-0.1	-0.13	-0.15	-0.077	0.58	
7	0	-0.18	-0.24	-0.19	-0.048	-0.18	-0.077	-0.15	-0.16	-0.11	- 0.00
9 -	0	0.68	-0.046	-0.21	-0.18	-0.12	-0.26	-0.17	-0.19	-0.12	
ا ک ح	0	-0.18	-0.31	-0.13	-0.11	-0.12	-0.2	-0.2	-0.44	0.034	
4 -	0	0.63	1	0.35	0.82	0.92	0.77	0.018	0.34	0.88	
m -	0	0.2	0.78	0.95	0.94	0.4	0.78	0.39	0.48	0.38	0.25
2 -	0	0.62	0.74	0.14	0.54	0.85	0.023	0.81	0.15	0.37	0.23
П -	0	0.58	0.96	0.41	0.33	0.85	0.57	0.89	0.52	0.59	
0 -	0	0.2	0.58	0.19	0.56	0.94	0.37	0.28	0.42	0.52	
	Ó	i	2	3	4	5	6	7	8	9	_ _

Schedule Length (# topics before orgs/lab)



0 50 100 150 200 250 State Index (Remaining Hours Increase with State Index)

PRIOR TO ORGS/LAB:

Optimal Free Time: 5 HOURS

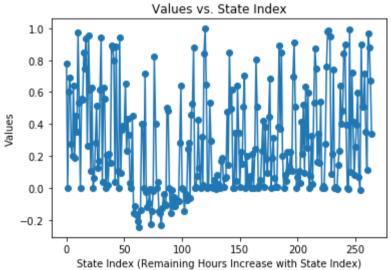
Optimal Schedule Length (without Orgs/Lab): 2 COMMITMENTS

FACTORING IN ORGS/LAB: Optimal Schedules:

Partially Optimal Schedules:

LAMBDA = 0.75
HBox(children=(IntProgress(value=0, max=100000), HTML(value='')))

	1	1	1	1	Max V alue	per State	1	1	1	1	
23	-0.011	0.9	0.51	0.51	0.72	0.35	0.11	0.97	0.88	0.67	-1.00
22	0	-0.0077	0.99	0.099	0.72	0.42	0.083	0.26	0.39	0.6	
72	0	0.0061	-0.0092	0.14	0.23	0.64	0.4	0.48	0.84	0.9	
2 -	0	0.01	0.0024	0.28	0.76	0.98	0.99	0.95	0.084	0.21	
61	0	0.072	0.33	0.027	0.53	0.87	0.75	0.17	0.34	0.15	- 0.75
82	0	0.066	0.1	0.03	0.42	0.21	0.52	0.095	0.63	0.25	
17	0	0.083	0.046	0.22	0.1	0.13	0.22	0.11	0.7	0.91	
16	0	0.19	0.31	0.063	0.0032	0.0081	0.38	0.89	0.37	0.85	
15	0	0.018	0.051	0.23	0.42	0.0073	0.38	0.18	0.097	0.45	
	0	0.2	0.028	0.056	0.077	-0.003	0.21	0.00067	0.24	0.81	- 0.50
Time (hrs)	0	-0.00011	0.35	0.044	0.64	0.35	0.0045	0.23	0.15	0.51	
Tim 12	0	0.17	0.19	0.0065	0.059	0.068	0.48	0.14	0.85	0.5	
uing 11	0	0.0032	0.53	0.3	0.0049	0.0024	0.05	-0.0052	0.079	0.026	
Remaining 9	0	0.0084	0.077	0.42	0.13	0.00094	0.32	0.017	0.84	1	
g 6	0	-0.0058	-0.14	-0.041	-0.075	0.24	0.28	-0.063	0.46	0.53	- 0.25
00 -	0	-0.16	-0.013	-0.11	-0.059	-0.12	-0.083	-0.074	-0.083	0.28	
7	0	0.038	-0.15	-0.16	-0.23	-0.13	-0.041	-0.099	-0.11	0.5	
9 -	0	0.71	-0.027	-0.12	-0.095	-0.009	-0.22	-0.15	-0.019	0.82	
20 -	0	0.45	-0.16	-0.12	-0.16	-0.17	-0.21	-0.25	-0.14	0.4	
4 -	0	0.94	0.092	0.36	0.39	0.39	0.65	0.23	0.43	0.33	- 0.00
Μ-	0	0.2	0.016	0.21	0.15	0.89	0.88	0.8	0.039	0.89	
2	0	0.06	0.29	0.51	0.12	0.17	0.62	0.94	0.035	0.63	
Т -	0	0.56	0.55	0.85	0.75	0.94	0.26	0.95	0.61	0.1	
0 -	0	0.6	0.69	0.27	0.2	0.64	0.19	0.45	0.35	0.97	
	Ó	i	2	3	4	5	6	7	8	9	<u>-</u>
				Schedule	e Length (# t	opics before	orgs/lab)				



PRIOR TO ORGS/LAB:

Optimal Free Time: 11 HOURS

Optimal Schedule Length (without Orgs/Lab): 9 COMMITMENTS

FACTORING IN ORGS/LAB: Optimal Schedules:

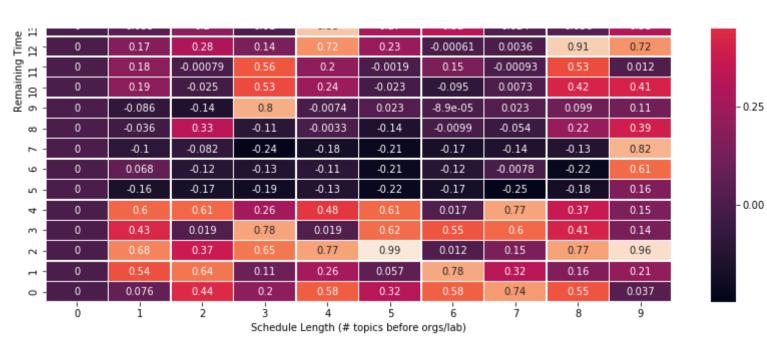
Partially Optimal Schedules:

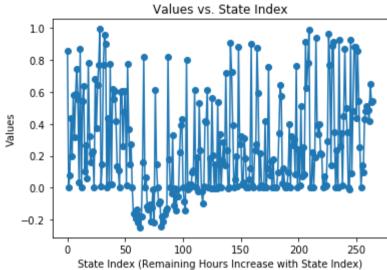
LAMBDA = 1
HBox(children=(IntProgress(value=0, max=100000), HTML(value='')))

		,				Ft-t-				,
	1	1	1	1	Max Value	per S tate	1	1	1	1
23	0.0015	0.14	0.098	0.43	0.48	0.51	0.49	0.42	0.65	0.53
22	0	-0.012	0.92	0.09	0.48	0.88	0.88	0.43	0.86	0.55
12	0	0.23	0.00064	0.93	0.26	0.34	0.017	0.45	0.87	0.5
8 -	0	0.071	0.0093	0.016	0.29	0.96	0.77	0.14	0.89	0.91
е Б	0	0.0051	0.22	0.0024	0.19	0.94	0.39	0.33	0.4	0.22
8 -	0	0.24	0.014	0.51	0.072	0.25	0.085	0.91	0.62	0.78
17	0	0.0021	0.012	0.096	0.4	0.0016	0.32	0.27	0.31	0.084
16	0	0.18	0.0019	0.0035	0.00058	0.1	0.34	0.64	0.57	0.025
55 -	0	0.22	0.017	1.4e-05	0.0062	0.41	0.00043	0.12	0.76	0.079
(hrs)	0	0.00026	0.055	0.12	0.9	0.015	0.0023	0.23	0.27	0.88
٤ []	0	0.053	0.2	0.01	0.88	0.27	0.32	0.024	0.036	0.31

- 0.75

- 1.00





Optimal Free Time: 3 HOURS

Optimal Schedule Length (without Orgs/Lab): 5 COMMITMENTS

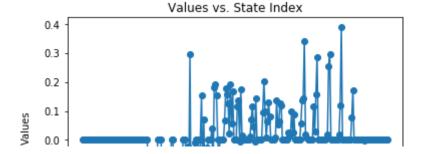
FACTORING IN ORGS/LAB: Optimal Schedules:

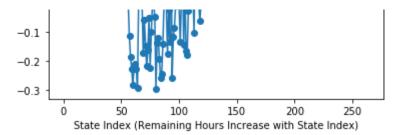
Partially Optimal Schedules:

INIT VALUES as zeros

LAMBDA = 0

	1	1	1	1	Max V alue	per S tate	1	1	1	1		
23	0.0024	0	0	0	0	0	0	0	0	0		- 1.00
72	0	-0.0028	0	0	0	0	0	0	0	0		
72	0	0.078	0.17	0	0	0	0	0	0	0		
22	0	0.02	0.12	0.39	0	0	0	0	0	0		
13	0	0.0044	0.02	0.25	0.29	0	0	0	0	0		- 0.75
81	0	0.12	0.029	0.16	0.28	0.0027	0	0	0	0		
17	0	0.058	0.14	0.14	0.34	0.02	0.008	0	0	0		
16	0	0.025	0.011	0.019	0.098	0.024	0.088	0	0	0		
15	0	0.01	0.14	0.054	0.065	0.13	0.12	0	0	0		
	0	0.094	0.2	0.098	0.0098	0.064	0.13	0.08	0	0		- 0.50
e (hi	0	0.012	0.07	0.12	0.047	-0.0063	0.14	0	0	0		
Tim 77	0	0.11	0.14	-0.0057	0.18	0.016	0.0073	0.00043	0	0		
Remaining 9	0	0.067	0.18	0.16	0.12	0.021	0.19	0.057	0.17	0		
mair 10	0	0.041	-0.1	0.18	0.19	0.15	4.7e-05	-0.061	0	0		- 0.25
Rel	0	-0.13	0.0029	0.15	-0.14	0.069	-0.16	-0.18	-0.027	0		0.23
∞ -	0	-0.18	-0.021	-0.13	0.3	-0.26	-0.11	-0.085	-0.0088	0		
7	0	-0.046	-0.3	-0.14	-0.12	-0.19	-0.26	-0.24	-0.14	0		
9 -	0	0	-0.17	-0.058	-0.15	-0.22	-0.16	-0.051	-0.22	-0.1		
<u>د</u> د	0	-0.11	-0.18	-0.23	-0.28	-0.22	-0.21	-0.22	-0.29	0		- 0.00
4 -	0	0	0	0	0	0	0	0	0	0		
m -	0	0	0	0	0	0	0	0	0	0		
2 -	0	0	0	0	0	0	0	0	0	0		
П -	0	0	0	0	0	0	0	0	0	0		
0 -	0	0	0	0	0	0	0	0	0	0		0.25
	Ó	i	2	3	4	5	6	7	8	9	,	_
				Schedule	e Length (# t	opics before	orgs/lab)					





Optimal Free Time: 21 HOURS

Optimal Schedule Length (without Orgs/Lab): 3 COMMITMENTS

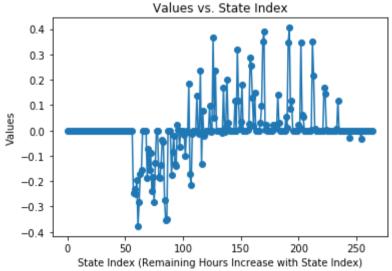
FACTORING IN ORGS/LAB: Optimal Schedules:

Partially Optimal Schedules:

LAMBDA = 0.5
HBox(children=(IntProgress(value=0, max=100000), HTML(value='')))

	1	1	1	1	Max V alue	per State	1	1	1	1	1.00
23	-0.031	0	0	0	0	0	0	0	0	0	- 1.00
72	0	-0.03	0	0	0	0	0	0	0	0	
21	0	0.0067	0.12	0	0	0	0	0	0	0	
50	0	0.17	0.14	0.0041	0	0	0	0	0	0	
19	0	0.0041	0.35	0.22	0.0065	0	0	0	0	0	- 0.75
- 18	0	0.021	0.011	0.35	0.063	0.054	0	0	0	0	
17	0	0.052	0.0093	0.35	0.41	0.086	0.12	0	0	0	
16	0	0.021	0.006	0.024	0.011	0.14	0.031	0	0	0	
15	0	0.032	0.1	0.35	0.39	0.0031	0.023	0	0	0	- 0.50
	0	0.0039	0.027	0.29	0.25	0.13	0.018	0.15	0	0	
e (hi 13	0	0.12	0.00018	0.32	0.12	0.0066	0.035	0.18	0	0	
Remaining Time (hrs) 9 10 11 12 13 14	0	-0.0086	0.17	-0.007	0.013	0.2	0.029	0.0011	0	0	
ning 11	0	0.096	-0.0051	-0.0029	0.37	0.048	0.24	-5e-05	0	0	0.25
mair 10	0	0.14	-0.0041	-0.012	0.24	-0.13	0.077	-0.02	-0.0095	0	- 0.25
Be -	0	-0.015	-0.099	-0.0017	-0.0064	0.18	-0.17	-0.22	-0.014	0	
ω -	0	-0.18	-0.086	-0.02	-0.13	-0.14	0.022	0.0046	-0.064	0	
7	0	-0.18	-0.19	-0.14	-0.035	-0.046	-0.27	-0.35	-0.35	0	
9 -	0	0	-0.19	-0.073	-0.16	-0.087	-0.24	-0.18	-0.28	-0.13	- 0.00
٠ م	0	-0.25	-0.23	-0.25	-0.19	-0.38	-0.28	-0.17	-0.16	0	
4 -	0	0	0	0	0	0	0	0	0	0	
m -	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0.25
П -	0	0	0	0	0	0	0	0	0	0	
0 -	0	0	0	0	0	0	0	0	0	0	
	ó	i	2	3	4	5	6	7	8	9	

Schedule Length (# topics before orgs/lab)



PRIOR TO ORGS/LAB:

Optimal Free Time: 18 HOURS

Optimal Schedule Length (without Orgs/Lab): 4 COMMITMENTS

FACTORING IN ORGS/LAB: Optimal Schedules:

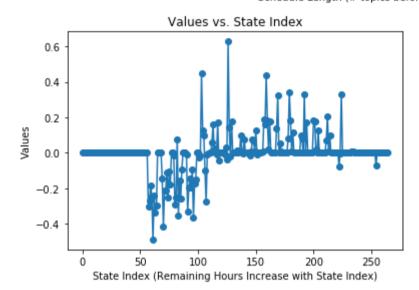
Partially Optimal Schedules:

LAMBDA = 0.75
HBox(children=(IntProgress(value=0, max=100000), HTML(value='')))

	1	1	1	1	Max V alue	per S tate	1	1	1	1
g -	-0.068	0	0	0	0	0	0	0	0	0
22 -	0	0.0049	0	0	0	0	0	0	0	0
77	0	0.0096	0.0063	0	0	0	0	0	0	0
8 -	0	-0.074	-0.0008	0.33	0	0	0	0	0	0
g -	0	0.073	0.21	0.0057	0.097	0	0	0	0	0
쬞 -	0	0.19	0.18	0.017	0.0031	0.12	0	0	0	0

- 1.00 - 0.75

17	0	0.1	0.08	0.0026	0.33	9.1e-05	0.17	0	0	0	
16	0	0.084	0.34	0.18	0.0082	0.12	0.00059	-1.1e-26	0	0	
55 -	0	0.0047	0.14	0.32	-5.4e-05	0.055	0.0073	0	0	0	
	0	0.0071	0.19	0.16	0.44	0.18	0.018	0.18	0	0	
lime (hrs) 12 13 14	0	-0.014	0.01	0.073	0.069	0.01	0.13	-0.0072	0	0	
Ē 23 -	0	0.014	0.0034	0.0061	0.097	0.0042	-0.0025	0.076	-3.4e-12	0	
ing 11	0	-0.0088	0.028	-0.037	0.63	0.14	-0.019	0.18	0	0	
Remaining 9 10 11	0	0.06	0.16	0.012	0.025	-0.0089	0.17	-0.043	0	0	
Ber	0	-0.026	-0.013	0.45	0.13	0.1	-0.1	-0.28	-0.01	-0.00042	
ω -	0	-0.01	-0.33	-0.19	-0.14	-0.2	-0.091	-0.36	-0.17	-0.15	
7	0	-0.017	-0.29	-0.25	0.076	-0.35	-0.16	-0.26	-0.092	0	
9 -	0	0	-0.15	-0.41	-0.21	-0.21	-0.11	-0.25	-0.11	-0.18	
ıς -	0	-0.3	-0.27	-0.18	-0.24	-0.49	-0.24	-0.34	-0.3	0	
4 -	0	0	0	0	0	0	0	0	0	0	
m -	0	0	0	0	0	0	0	0	0	0	ı
2 -	0	0	0	0	0	0	0	0	0	0	ı
П -	0	0	0	0	0	0	0	0	0	0	ı
0 -	0	0	0	0	0	0	0	0	0	0	
	Ó	i	2	3	4	5	6	7	8	9	_
				Schedule	e Length (# t	opics before	orgs/lab)				



Optimal Free Time: 12 HOURS

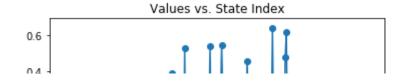
Optimal Schedule Length (without Orgs/Lab): 4 COMMITMENTS

FACTORING IN ORGS/LAB: Optimal Schedules:

Partially Optimal Schedules:

LAMBDA = 1HBox(children=(IntProgress(value=0, max=100000), HTML(value='')))

	1	1	1	1	Max V alue	per S tate	1	1	1	1	1.00
- 33	-0.003	0	0	0	0	0	0	0	0	0	- 1.00
22 -	0	0.00053	0	0	0	0	0	0	0	0	
21	0	-9.3e-05	-0.09	0	0	0	0	0	0	0	
8 -	0	0.043	-0.091	0.3	0	0	0	0	0	0	
5 ·	0	0.14	0.01	0.19	0.13	0	0	0	0	0	- 0.75
85 -	0	-1.1e-05	0.022	0.12	0.021	0.00092	0	0	0	0	
17	0	0.1	0.0023	0.48	0.61	0.0013	0.057	0	0	0	
16	0	0.07	0.13	0.64	0.053	0.031	0.0014	0	0	0	
55 ·	0	0.12	0.0013	0.037	0.35	0.00048	0.072	0	0	0	- 0.50
(ST)	0	0.063	0.0012	0.45	0.28	0.0011	0.21	0.13	0	0	
e (h	0	-0.015	0.02	0.26	-0.0016	0.25	0.11	-0.0011	0	0	
Time (hrs) 12 13 14 '	0	-0.074	0.25	0.55	0.27	-0.0011	0.22	0.017	-0.022	0	
Remaining 9 10 11 1	0	0.085	0.34	0.17	0.54	0.011	0.0092	0.14	0.24	0	- 0.25
mair 10	0	0.073	0.27	-0.0062	0.0015	0.18	0.031	0.23	0.25	0	
9 e -	0	0.11	-0.23	-0.18	0.53	-0.011	-0.0038	0.17	-0.16	0	
∞ -	0	-0.072	-0.24	-0.13	0.39	-0.013	-0.018	-0.24	-0.0064	0	
7	0	0.077	-0.32	-0.28	-0.17	-0.19	-0.33	-0.22	-0.23	0	- 0.00
9 -	0	0	-0.29	-0.31	-0.25	-0.31	-0.46	-0.29	-0.2	-0.18	
ru -	0	0	-0.27	-0.29	-0.24	-0.2	-0.4	-0.2	-0.44	-0.2	
4 -	0	0	0	0	0	0	0	0	0	0	
m -	0	0	0	0	0	0	0	0	0	0	0.25
2 -	0	0	0	0	0	0	0	0	0	0	
-	0	0	0	0	0	0	0	0	0	0	
0 -	0	0	0	0	0	0	0	0	0	0	
	ò	i	2	3 Sebadula	4	5 opics before	6 6	7	8	9	



```
0.2 - 0.0 - -0.2 - -0.4 - 0 50 100 150 200 250 State Index (Remaining Hours Increase with State Index)
```

Optimal Free Time: 17 HOURS

Optimal Schedule Length (without Orgs/Lab): 3 COMMITMENTS

FACTORING IN ORGS/I AR.

▼ Save Variables

```
with open('TD_init_values.pickle', 'wb') as td_results1:
    pickle.dump(TD_init_values, td_results1)
with open('TD_init_eps.pickle', 'wb') as td_results2:
    pickle.dump(TD_init_eps, td_results2)
with open('TD_init_ep_rewards.pickle', 'wb') as td_results3:
    pickle.dump(TD_init_ep_rewards, td_results3)
```