inventory_beer_game_implementation_new

August 20, 2023

1 Reinforcement Learning Project - Inventory beer game

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import random
import sys
```

1.0.1 Setup

[ ]:
```
# Supply Chain and its agents
supply_chain = {'level 0': 'Customer',
                'level 1': 'Retailer',
                'level 2': 'Distributor',
                'level 3': 'Manufacturer'}

agents = [supply_chain[i] for i in list(supply_chain.keys())[1:]]


# All possible coded inventory positions of agents and the respective state␣
 ↪pairs (5 states with 25 state pairs)
states = np.arange(start=1, stop=6)
state_pairs = [(i, j, k) for i in states for j in states for k in states]

# y-value in ordering rule x+y with x equal to the demand from the downstream␣
 ↪agent and the respective action pairs (4 actions with 16 action pairs)
actions = np.arange(stop=4)
action_pairs = [(i, j, k) for i in actions for j in actions for k in actions]


# Initial matrix with Q-values (minimization -> high values)
max_value = 1000
Q_values = np.full(shape=(len(state_pairs), len(action_pairs)),␣
 ↪fill_value=max_value)
df_Q_values = pd.DataFrame(data=Q_values, index=state_pairs,␣
 ↪columns=action_pairs)
display(df_Q_values.head())
```

```
           (0, 0, 0)  (0, 0, 1)  (0, 0, 2)  (0, 0, 3)  (0, 1, 0)  (0, 1, 1)  \
(1, 1, 1)       1000       1000       1000       1000       1000       1000
(1, 1, 2)       1000       1000       1000       1000       1000       1000
(1, 1, 3)       1000       1000       1000       1000       1000       1000
(1, 1, 4)       1000       1000       1000       1000       1000       1000
(1, 1, 5)       1000       1000       1000       1000       1000       1000

           (0, 1, 2)  (0, 1, 3)  (0, 2, 0)  (0, 2, 1)  …  (3, 1, 2)  \
(1, 1, 1)       1000       1000       1000       1000  …       1000
(1, 1, 2)       1000       1000       1000       1000  …       1000
(1, 1, 3)       1000       1000       1000       1000  …       1000
(1, 1, 4)       1000       1000       1000       1000  …       1000
(1, 1, 5)       1000       1000       1000       1000  …       1000

           (3, 1, 3)  (3, 2, 0)  (3, 2, 1)  (3, 2, 2)  (3, 2, 3)  (3, 3, 0)  \
(1, 1, 1)       1000       1000       1000       1000       1000       1000
(1, 1, 2)       1000       1000       1000       1000       1000       1000
(1, 1, 3)       1000       1000       1000       1000       1000       1000
(1, 1, 4)       1000       1000       1000       1000       1000       1000
(1, 1, 5)       1000       1000       1000       1000       1000       1000

           (3, 3, 1)  (3, 3, 2)  (3, 3, 3)
(1, 1, 1)       1000       1000       1000
(1, 1, 2)       1000       1000       1000
(1, 1, 3)       1000       1000       1000
(1, 1, 4)       1000       1000       1000
(1, 1, 5)       1000       1000       1000

[5 rows x 64 columns]
```

[ ]:
```python
# Define parameters
iteration = 1
max_iteration = 20000
T = 10
gamma = 0.9 #low: faster convergence! less variance
alpha = 0.17 #high: faster updating q-values (more weight to new observations)␣
 ↪and big variance in q-value updates
sigma = 0.01
proba_exploitation = 0.02
start_exploitation = proba_exploitation

# Set up Lists to store the results of each iteration/episode
S = [list(np.repeat(0, len(agents))) for i in range(T+1)]
CS = [list(np.repeat(0, len(agents))) for i in range(T+1)]
D = [list(np.repeat(0, len(agents))) for i in range(T+1)]
O = [list(np.repeat(0, len(agents))) for i in range(T+1)]
x = [list(np.repeat(0, len(agents))) for i in range(T+1)]
```

```
y = [list(np.repeat(0, len(agents))) for i in range(T+1)]
r = [list(np.repeat(0, len(agents))) for i in range(T+1)]
R = [0 for i in range(T)]
G = [0 for i in range(T)]
q = [0 for i in range(T)]
Q = [0 for i in range(max_iteration)]
```

Meaning of the variables - S: (start-)inventory positions/states per agent and time step t - CS: coded states per agent and time step t - O: ordering size per agent and time step t - D: distribution amount per agent and time step t - x: demand from downstream level per agent and time step t - y: action per agent and time step t - r: reward per agent and time step t - R: reward of the supply chain per time step t - Return G: total discounted rewards per time step t - q: measure the decrease of the Q-value per time step t - Q: measure the mean decrease of Q-values per episode to indicate convergence

[ ]:
```
# helperfunctions
# function to convert real states to coded states
def coded_state(inventory):
    if inventory < -5:
        return 1
    elif inventory < 0:
        return 2
    elif inventory < 5:
        return 3
    elif inventory < 12:
        return 4
    else:
        return 5


# function to view episodes
def fun_episode(S, CS, D, O, x, y, r, head=True):
    time_steps = ['t='+str(i) for i in np.arange(start=0, stop=T+1)]

    df = pd.DataFrame({'Inventory/States S': [tuple(i) for i in S],
                   'Coded states CS': [tuple(i) for i in CS],
                   'Demand x': [tuple(i) for i in x],
                   'Distribution amount D': [tuple(i) for i in D],
                   'Action y': [tuple(i) for i in y],
                   'Ordering size': [tuple(i) for i in O],
                   'Costs r': [tuple(i) for i in r]},
                   index=time_steps)
    df.index.name = 'Time'
    if head == True:
        return display(df.head())
    else: return display(df)
```

```
# function to define the starting states of the supply chain
def fun_start_state(how='value', inv=10):
    # select own start values for all agents
    if how == 'value':
        for agent in range(len(agents)):
            S[0][agent] = inv
            CS[0][agent] = coded_state(S[0][agent])

    # each episode has a change of 50% to start with high (10) or with low (1) inventory for all agents
    elif how == 'high/low':
        starting_state = random.choices(population=['high', 'low'], weights=[0.5, 0.5])[0]
        if starting_state == 'high':
            for agent in range(len(agents)):
                S[0][agent] = 10
                CS[0][agent] = coded_state(S[0][agent])
        else:
            for agent in range(len(agents)):
                S[0][agent] = 1
                CS[0][agent] = coded_state(S[0][agent])

    # random starting positions (-10 to 15) of agents for each episode
    elif how == 'random':
        for agent in range(len(agents)):
            S[0][agent] = random.choices(np.arange(start=-10, stop=16))[0]
            CS[0][agent] = coded_state(S[0][agent])


# function to measure the Q-value decrease
def fun_q_decrease(alpha, q_value, G):
    new_q_value = (1-alpha) * q_value + alpha * G
    decrease = (q_value - new_q_value) / q_value
    return new_q_value, decrease


# function to visualize the Q-vlaue decrease
def plot_q_decrease(Q, iteration):
    plt.figure(figsize=(12,4))
    plt.plot(Q)
    plt.xlim(0, iteration)
    plt.ylim(-0.1, 1.1)
    plt.xlabel('episode/iteration')
    plt.ylabel('Q-value decrease')
    plt.title('Q-value decrease per episode', size=18)
    return plt.show()
```

[ ]:
```
# Three possible options to set starting states
fun_start_state(how='value', inv=10)
print('Inventory state S: {}'.format(S))
print('Coded state S: {}\n'.format(CS))

fun_start_state(how='high/low')
print('Inventory state S: {}'.format(S))
print('Coded state S: {}\n'.format(CS))

fun_start_state(how='random')
print('Inventory state S: {}'.format(S))
print('Coded state S: {}\n'.format(CS))
```

```
Inventory state S: [[10, 10, 10], [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]]
Coded state S: [[4, 4, 4], [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]]

Inventory state S: [[1, 1, 1], [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]]
Coded state S: [[3, 3, 3], [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]]

Inventory state S: [[-8, 6, -9], [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]]
Coded state S: [[1, 4, 1], [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]]
```

[ ]:
```
#%%capture
# Main function

while iteration <= max_iteration:
    if iteration in [1, 10, 100, 1000, 10000, 25000, 50000, 75000, 100000,␣
 ↪max_iteration]:
        print('iteration {}'.format(iteration))

    #print('--------------------episode {}-------------------'.format(iteration))
    #print('Exploitation probability: {}\n'.format(proba_exploitation))
    t=0

    # random starting position of agents for each episode
    fun_start_state(how='high/low') #option 1: ('value', inv=12); option 2:␣
 ↪'high/low'; option 3: 'random'

    while t < T:
        #print('-----------------time step t={}----------------'.format(t))
```

```
        # view state and coded state
        state = tuple(S[t])
        c_state = tuple(CS[t])
        #print('State S: {} and coded state CS: {}'.format(state, c_state))

        for agent in range(len(agents)):
            level = supply_chain['level ' + str(agent+1)]
            #print('------------------{}------------------'.format(level))

        # step 1: receive the new demand of the downstream agent
            if level == 'Retailer':
                x[t][agent] = np.random.randint(low=0, high=15)
            else:
                x[t][agent] = O[t-1][agent-1]

            # Add negative inventory (=demand of previous time steps) to the␣
↪new demand
            inventory = S[t][agent]
            if inventory >= 0:
                #print('Demand x from downstream ({}) at t-1: {}'.
↪format(supply_chain['level ' + str(agent)], x[t][agent]))
                pass
            else:
                #print('Demand x from downstream ({}) at t-1 + demand of␣
↪previous time steps:'.format(supply_chain['level ' + str(agent)]))
                #print(    '{} + {} = {}'.format(x[t][agent], np.
↪abs(inventory), x[t][agent] + np.abs(inventory)))
                x[t][agent] += np.abs(inventory)

        # step 2: fulfill order of downstream agent from onhand inventory and␣
↪calculate possible backlog costs
            if (inventory >= 0) & (level != agents[-1]):
                D[t][agent] = min(x[t][agent], max(inventory, D[t-1][agent+1]))␣
↪#distribution quantity is the demand or the maximum of inventory and last␣
↪received order
                #print('Distribution size D: {}'.format(D[t][agent]))
            elif level != agents[-1]:
                D[t][agent] = min(x[t][agent], D[t-1][agent+1]) #negative␣
↪inventory is demand of previous time steps and still needs to be complied
                #print('Distribution size D (delivery of upstream agent in t-1):
↪ {}'.format(D[t][agent])) #(max of demand and last received order)
            elif (inventory >= 0) & (level == agents[-1]):
                D[t][agent] = min(x[t][agent], max(inventory, O[t-1][agent]))␣
↪#last agent receives his own order out of warehouse
                #print('Distribution size D: {}'.format(D[t][agent]))
```

```
            else:
                D[t][agent] = min(x[t][agent], O[t-1][agent])
                #print('Distribution size D (delivery of warehouse): {}'.
↪format(D[t][agent]))

            backlog = x[t][agent] - D[t][agent] #penalty/backlog costs␣
↪(previous backlogs included in demand if inventory is negative)
            #print('Backlog size: {}'.format(backlog))

        # step 3: placing order for stock replenishment
            # define best action for agent and select it with initially very␣
↪small probability (first exploration and mostly random choices)
            best_action = df_Q_values.iloc[df_Q_values.index.
↪get_loc(tuple(CS[t]))].idxmin()[agent]
            #print('Best action y*: {}'.format(best_action))

            y[t][agent] = random.choices([best_action] + list(actions),␣
↪weights=[proba_exploitation] + list(np.repeat((1 - proba_exploitation) /␣
↪len(actions), len(actions))))[0]
            #print('Action y: {}'.format(y[t][agent]))

            if inventory >= 0:
                O[t][agent] = x[t][agent] + y[t][agent]
            else: O[t][agent] = x[t][agent] + inventory + y[t][agent] #subtract␣
↪negative inventory (demand of previous time steps) again - has been ordered␣
↪already
            #print('Ordering size O: {}\n'.format(O[t][agent]))

        # step 4: previous orders are received from the upstream agent (update␣
↪states for t+1)
        #print('UPDATING STATES AND CALCULATING COSTS PER AGENT')
        for agent in range(len(agents)):
            level = supply_chain['level ' + str(agent+1)]
            #print('-------------------{}-------------------'.format(level))

            # update inventory with demand of t
            if  S[t][agent] >= 0:
                inventory =  S[t][agent] - x[t][agent]
            else: inventory = - x[t][agent] #demand x contains negative␣
↪inventory + new demand = new inventory

            # update state t+1 with inventory + received order
            if level != agents[-1]:
                S[t+1][agent] = inventory + D[t][agent+1]
                #print('Inventory {} after receiving order +{} of t: {}'.
↪format(inventory, D[t][agent+1], S[t+1][agent]))
```

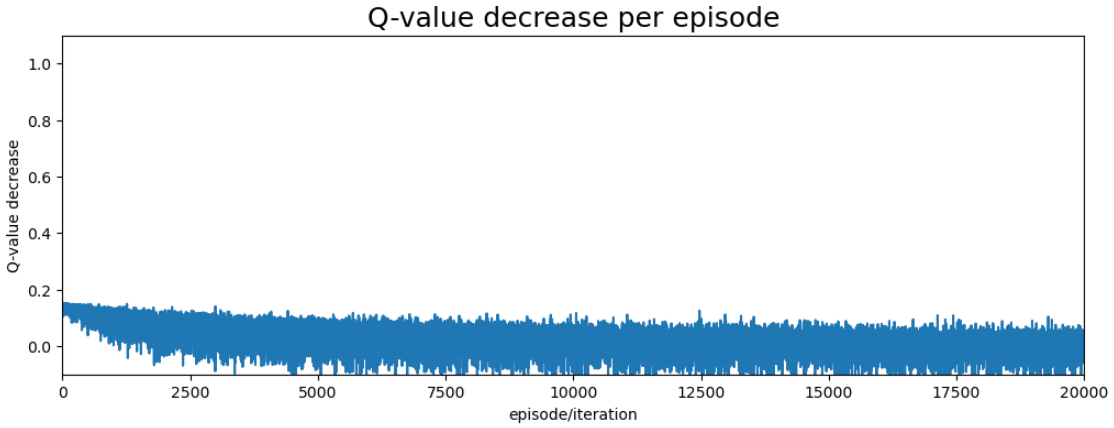
```
            # for last agent in supply chain: update state t+1 with inventory + order of t-1 (delivers from warehouse)
            else:
                S[t+1][agent] = inventory + O[t-1][agent]
                #print('Inventory {} receiving order +{} of t-1: {}'.format(inventory, O[t-1][agent], S[t+1][agent]))

            # update coded states
            CS[t+1][agent] = coded_state(S[t+1][agent])

            # calculate agent's costs (onhand inventory holding costs + penalty costs)
            r[t][agent] = 1 * max(S[t+1][agent], 0) + 2 * (x[t][agent] - D[t][agent]) #backlog
            #print('Costs r: {}\n'.format(r[t][agent]))

        # calculate the total supply chain costs in t
        action = tuple(y[t])
        R[t] = np.sum(r[t])
        #print('Supply Chain costs R in state {} with action {} at t={}: {}'.format(state, action, t, R[t]))

        # increase time step t
        t += 1
        #print('\n\n')

    # view last episode
    #fun_episode(S, CS, D, O, x, y, r, head=False)

    # Loop from T to t=0 to calculate immediate rewards and returns
    #print('-----Rewards and Total discounted returns-----')
    for t in np.arange(start=0, stop=T)[::-1]:
        # view immediate reward of each visited state-action-pair
        #print('Immediate reward R in t={}: {}'.format(t, R[t]))

        # calculate the return G: total discounted rewards
        G[t] = R[t] + np.sum(R[t+1:] * np.array([gamma**i for i in np.arange(start=1, stop=T-t)]))

    # view returns of all state-action-pairs
    #print('\nTotal discounted rewards (Return G): \n{}\n'.format(G))

    # update all visited Q-values
    #print('---------------UPDATING Q-VALUES--------------')
    for t in range(T):
```

```
        state = tuple(CS[t])
        action = tuple(y[t])

        # get current Q-value
        q_value = df_Q_values.iloc[df_Q_values.index.get_loc(state),␣
↪df_Q_values.columns.get_loc(action)]
        #print('t={}: state: {}, action: {}, old Q-value: {}, return: {}'.
↪format(t, state, action, q_value, G[t]))

        # update Q-values according to equation 12 in the paper (slide 30␣
↪TD-learning script)
        df_Q_values.iloc[df_Q_values.index.get_loc(state), df_Q_values.columns.
↪get_loc(action)] = (1-alpha) * q_value + alpha * G[t] #equal to: q_value +␣
↪alpha * (G[t] - q_value)
        new_q_value, decrease = fun_q_decrease(alpha, q_value, G[t])
        q[t] = decrease
        #print('New Q-value: {}\n'.format(new_q_value))

    # calculate mean Q-value decrease of current episode
    Q[iteration-1] = np.mean(q)
    #print('Mean Q-value decrease of all visited state-action-pairs in this␣
↪episode: {}'.format(Q[iteration-1]))

    # Check if the convergence criterion has been reached
    if (iteration > 100):
        if (np.mean(np.abs(Q[iteration-20:iteration])) < sigma):
            print('Convergence at episode {}!'.format(iteration))
            break

    # increase exploitation probability linearly
    proba_exploitation += (0.95 - start_exploitation) / max_iteration
    #print('\n\n')

    # start with next iteration
    iteration += 1

# round dataframe with Q-values
df_Q_values = np.round(df_Q_values, 2)

# view new Q-value table
display(df_Q_values)

# plot decrease of Q-vlaues
plot_q_decrease(Q, iteration)
```

```
iteration 1
iteration 10
```

```
iteration 100
iteration 1000
iteration 10000
iteration 20000
           (0, 0, 0)  (0, 0, 1)  (0, 0, 2)  (0, 0, 3)  (0, 1, 0)  (0, 1, 1)  \
(1, 1, 1)     255.82     260.85     264.40     269.18     260.03     277.16
(1, 1, 2)     254.61     236.41     239.92     269.46     250.03     249.81
(1, 1, 3)     350.36     305.21     330.47     316.68     313.40     321.50
(1, 1, 4)     162.55     170.74     154.90     158.94     198.08     172.56
(1, 1, 5)     838.50    1000.00     616.05     649.73     398.60     853.96
…               …          …          …          …          …
(5, 5, 1)    1000.00    1000.00    1000.00    1000.00    1000.00    1000.00
(5, 5, 2)    1000.00    1000.00    1000.00     845.29     853.23     845.49
(5, 5, 3)     853.54    1000.00    1000.00     845.13     361.26    1000.00
(5, 5, 4)     525.52     447.54     127.14     373.18     734.14     843.91
(5, 5, 5)     615.40     726.76     716.11     863.21    1000.00     608.72

           (0, 1, 2)  (0, 1, 3)  (0, 2, 0)  (0, 2, 1)  …  (3, 1, 2)  \
(1, 1, 1)     251.65     280.95     271.76     262.62  …     261.07
(1, 1, 2)     252.26     242.05     280.27     242.00  …     234.08
(1, 1, 3)     297.45     313.75     313.99     313.72  …     270.49
(1, 1, 4)     159.02     191.82     169.44     178.61  …     165.38
(1, 1, 5)     518.03     614.45     445.14    1000.00  …    1000.00
…               …          …          …          …  …          …
(5, 5, 1)    1000.00    1000.00    1000.00    1000.00  …    1000.00
(5, 5, 2)     844.63     595.74    1000.00     840.93  …    1000.00
(5, 5, 3)    1000.00    1000.00     736.69    1000.00  …     737.64
(5, 5, 4)     731.36     534.21     541.91     717.97  …     708.45
(5, 5, 5)     505.21     844.96     508.05     114.39  …     840.54

           (3, 1, 3)  (3, 2, 0)  (3, 2, 1)  (3, 2, 2)  (3, 2, 3)  (3, 3, 0)  \
(1, 1, 1)     281.41     247.55     240.75     238.69     243.10     245.80
(1, 1, 2)     169.79     248.21     239.44     251.40     255.54     291.14
(1, 1, 3)     301.61     316.38     304.80     313.20     319.78     313.71
(1, 1, 4)     157.00     163.90     156.89     166.88     184.33     219.42
(1, 1, 5)     732.21     715.19    1000.00     843.77     632.63     518.33
…               …          …          …          …          …
(5, 5, 1)    1000.00    1000.00    1000.00    1000.00    1000.00    1000.00
(5, 5, 2)    1000.00    1000.00    1000.00    1000.00    1000.00    1000.00
(5, 5, 3)    1000.00     856.80    1000.00    1000.00    1000.00    1000.00
(5, 5, 4)     607.96     739.12     517.82     835.44     633.29     611.86
(5, 5, 5)     601.92     840.54     166.89     526.54     396.57     611.68

           (3, 3, 1)  (3, 3, 2)  (3, 3, 3)
(1, 1, 1)     232.53     263.27     231.36
(1, 1, 2)     236.85     274.13     223.66
(1, 1, 3)     342.89     330.24     306.10
```

```
(1, 1, 4)     232.80     187.33     199.40
(1, 1, 5)     452.88     615.24     263.96
…                …          …          …
(5, 5, 1)    1000.00    1000.00    1000.00
(5, 5, 2)    1000.00    1000.00    1000.00
(5, 5, 3)    1000.00    1000.00    1000.00
(5, 5, 4)     528.83     356.39     546.01
(5, 5, 5)     221.81     524.94     851.06

[125 rows x 64 columns]
```

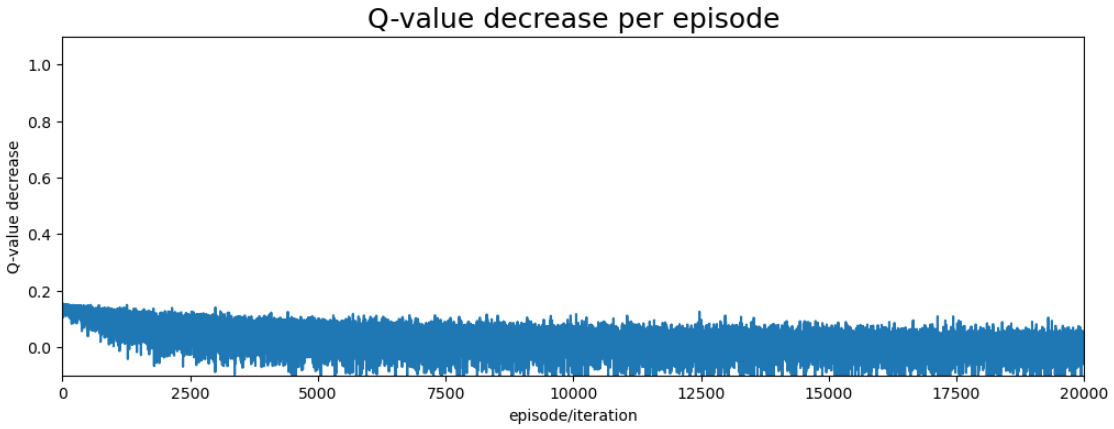


1.0.2 View new Q-values

[ ]:
```
# Percentage of updated Q-values
num_pairs = len(df_Q_values.index) * len(df_Q_values.columns)
print('Number of state-action-pairs: {}'.format(num_pairs))
print('Percentage of updated Q-values: {} %\n'.format(np.round(np.sum(np.
  ↪sum(df_Q_values != max_value)) / num_pairs * 100, 4)))

visits = max_iteration * T
print('Number of visits with max_iteration={} and T={}: {}'.
  ↪format(max_iteration, T, visits))
print('Average amount of visits per state-action-pair: {}\n'.format(np.
  ↪round(visits / num_pairs, 2)))

# plot decrease of Q-vlaues
print('Last twenty Q-value decreases:\n{}'.format(np.round(np.
  ↪abs(Q[iteration-20:iteration]), 4)))
print('Mean of last twenty Q-value decreases: {}'.format(np.round(np.mean(np.
  ↪abs(Q[iteration-20:iteration])), 4)))

plot_q_decrease(Q, iteration)
```

```
df_Q_values
```

```
Number of state-action-pairs: 8000
Percentage of updated Q-values: 67.8875 %

Number of visits with max_iteration=20000 and T=35: 700000
Average amount of visits per state-action-pair: 87.5

Last twenty Q-value decreases:
[0.0586 0.0599 0.028  0.0367 0.0209 0.0196 0.005  0.0124 0.0449 0.0221
 0.0077 0.0131 0.0145 0.0126 0.0211 0.0354 0.0315 0.0041 0.0583]
Mean of last twenty Q-value decreases: 0.0267
```



[ ]:

	(0, 0, 0)	(0, 0, 1)	(0, 0, 2)	(0, 0, 3)	(0, 1, 0)	(0, 1, 1)	\
(1, 1, 1)	255.82	260.85	264.40	269.18	260.03	277.16	
(1, 1, 2)	254.61	236.41	239.92	269.46	250.03	249.81	
(1, 1, 3)	350.36	305.21	330.47	316.68	313.40	321.50	
(1, 1, 4)	162.55	170.74	154.90	158.94	198.08	172.56	
(1, 1, 5)	838.50	1000.00	616.05	649.73	398.60	853.96	
…	…	…	…	…	…	…	
(5, 5, 1)	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	
(5, 5, 2)	1000.00	1000.00	1000.00	845.29	853.23	845.49	
(5, 5, 3)	853.54	1000.00	1000.00	845.13	361.26	1000.00	
(5, 5, 4)	525.52	447.54	127.14	373.18	734.14	843.91	
(5, 5, 5)	615.40	726.76	716.11	863.21	1000.00	608.72	

	(0, 1, 2)	(0, 1, 3)	(0, 2, 0)	(0, 2, 1)	…	(3, 1, 2)	\
(1, 1, 1)	251.65	280.95	271.76	262.62	…	261.07	
(1, 1, 2)	252.26	242.05	280.27	242.00	…	234.08	
(1, 1, 3)	297.45	313.75	313.99	313.72	…	270.49	
(1, 1, 4)	159.02	191.82	169.44	178.61	…	165.38	

```
(1, 1, 5)     518.03     614.45     445.14    1000.00  …    1000.00
…               …          …          …         …        …
(5, 5, 1)    1000.00    1000.00    1000.00    1000.00  …    1000.00
(5, 5, 2)     844.63     595.74    1000.00     840.93  …    1000.00
(5, 5, 3)    1000.00    1000.00     736.69    1000.00  …     737.64
(5, 5, 4)     731.36     534.21     541.91     717.97  …     708.45
(5, 5, 5)     505.21     844.96     508.05     114.39  …     840.54

            (3, 1, 3)  (3, 2, 0)  (3, 2, 1)  (3, 2, 2)  (3, 2, 3)  (3, 3, 0)  \
(1, 1, 1)     281.41     247.55     240.75     238.69     243.10     245.80
(1, 1, 2)     169.79     248.21     239.44     251.40     255.54     291.14
(1, 1, 3)     301.61     316.38     304.80     313.20     319.78     313.71
(1, 1, 4)     157.00     163.90     156.89     166.88     184.33     219.42
(1, 1, 5)     732.21     715.19    1000.00     843.77     632.63     518.33
…               …          …          …          …          …          …
(5, 5, 1)    1000.00    1000.00    1000.00    1000.00    1000.00    1000.00
(5, 5, 2)    1000.00    1000.00    1000.00    1000.00    1000.00    1000.00
(5, 5, 3)    1000.00     856.80    1000.00    1000.00    1000.00    1000.00
(5, 5, 4)     607.96     739.12     517.82     835.44     633.29     611.86
(5, 5, 5)     601.92     840.54     166.89     526.54     396.57     611.68

            (3, 3, 1)  (3, 3, 2)  (3, 3, 3)
(1, 1, 1)     232.53     263.27     231.36
(1, 1, 2)     236.85     274.13     223.66
(1, 1, 3)     342.89     330.24     306.10
(1, 1, 4)     232.80     187.33     199.40
(1, 1, 5)     452.88     615.24     263.96
…               …          …          …
(5, 5, 1)    1000.00    1000.00    1000.00
(5, 5, 2)    1000.00    1000.00    1000.00
(5, 5, 3)    1000.00    1000.00    1000.00
(5, 5, 4)     528.83     356.39     546.01
(5, 5, 5)     221.81     524.94     851.06

[125 rows x 64 columns]
```

Some Q-values stay unupdated because they represent unrealistic scenarios, where the downstream agents have much higher stock level than the upstream agents. That makes them less relevant!

1.0.3 Test performance of the Reinforcement learning ordering mechanism (RLOM) on main and test problems

[ ]:
```
# Main test problem
main_set = {'customer_demand':␣
 ↪[15,10,8,14,9,3,13,2,13,11,3,4,6,11,15,12,15,4,12,3,13,10,15,15,3,11,1,13,10,10,0,0,8,0,14]}


# Test problem 1
```

```
set1 = {'customer_demand':␣
 ↪[5,14,14,13,2,9,5,9,14,14,12,7,5,1,13,3,12,4,0,15,11,10,6,0,6,6,5,11,8,4,4,12,13,8,12]}

# Test problem 2
set2 = {'customer_demand':␣
 ↪[13,13,12,10,14,13,13,10,2,12,11,9,11,3,7,6,12,12,3,10,3,9,4,15,12,7,15,5,1,15,11,9,14,0,4]

# Own test problem
set3 = {'customer_demand': list(np.random.randint(low=0, high=15, size=35))}
```

[ ]:
```
def main_function(problem_set, policy, t=0, T=len(main_set['customer_demand'])):
    # set starting position of agents
    fun_start_state(how='value', inv=10)

    while t < T:
        print('-----------------time step t={}----------------'.format(t))

        # view state and coded state
        state = tuple(S[t])
        c_state = tuple(CS[t])
        print('State S: {} and coded state CS: {}'.format(state, c_state))

        for agent in range(len(agents)):
            level = supply_chain['level ' + str(agent+1)]
            print('------------------{}------------------'.format(level))

        # step 1: receive the new demand of the downstream agent
            if level == 'Retailer':
                x[t][agent] = problem_set['customer_demand'][t]
            else:
                x[t][agent] = O[t-1][agent-1]

            # Add negative inventory (=demand of previous time steps) to the␣
 ↪new demand
            inventory = S[t][agent]
            if inventory >= 0:
                print('Demand x from downstream ({}) at t-1: {}'.
 ↪format(supply_chain['level ' + str(agent)], x[t][agent]))
                pass
            else:
                print('Demand x from downstream ({}) at t-1 + demand of␣
 ↪previous time steps:'.format(supply_chain['level ' + str(agent)]))
                print(    '{} + {} = {}'.format(x[t][agent], np.abs(inventory),␣
 ↪x[t][agent] + np.abs(inventory)))
                x[t][agent] += np.abs(inventory)
```

```
        # step 2: fulfill order of downstream agent from onhand inventory and␣
↪calculate possible backlog costs
            if (inventory >= 0) & (level != agents[-1]):
                D[t][agent] = min(x[t][agent], max(inventory, D[t-1][agent+1]))␣
↪#distribution quantity is the demand or the maximum of inventory and last␣
↪received order
                print('Distribution size D: {}'.format(D[t][agent]))
            elif level != agents[-1]:
                D[t][agent] = min(x[t][agent], D[t-1][agent+1]) #negative␣
↪inventory is demand of previous time steps and still needs to be complied
                print('Distribution size D (delivery of upstream agent in t-1):␣
↪{}'.format(D[t][agent])) #(max of demand and last received order)
            elif (inventory >= 0) & (level == agents[-1]):
                D[t][agent] = min(x[t][agent], max(inventory, O[t-1][agent]))␣
↪#last agent receives his own order out of warehouse
                print('Distribution size D: {}'.format(D[t][agent]))
            else:
                D[t][agent] = min(x[t][agent], O[t-1][agent])
                print('Distribution size D (delivery of warehouse): {}'.
↪format(D[t][agent]))

            backlog = x[t][agent] - D[t][agent] #penalty/backlog costs␣
↪(previous backlogs included in demand if inventory is negative)
            print('Backlog size: {}'.format(backlog))


        # step 3: placing order for stock replenishment
            if policy == 'optimum':
                y[t][agent] = opt_policy[tuple(CS[t])][agent]
            elif policy == 'zero':
                y[t][agent] = 0
            else: y[t][agent] = np.random.randint(4)
            print('Action y: {}'.format(y[t][agent]))

            if inventory >= 0:
                O[t][agent] = x[t][agent] + y[t][agent]
            else: O[t][agent] = x[t][agent] + inventory + y[t][agent] #subtract␣
↪negative inventory (demand of previous time steps) again - has been ordered␣
↪already
            print('Ordering size O: {}\n'.format(O[t][agent]))

        # step 4: previous orders are received from the upstream agent (update␣
↪states for t+1)
        print('UPDATING STATES AND CALCULATING COSTS PER AGENT')
        for agent in range(len(agents)):
            level = supply_chain['level ' + str(agent+1)]
```

```
            print('------------------{}------------------'.format(level))

            # update inventory with demand of t
            if  S[t][agent] >= 0:
                inventory =  S[t][agent] - x[t][agent]
            else: inventory = - x[t][agent] #demand x contains negative␣
↪inventory + new demand = new inventory

            # update state t+1 with inventory + received order
            if level != agents[-1]:
                S[t+1][agent] = inventory + D[t][agent+1]
                print('Inventory {} after receiving order +{} of t: {}'.
↪format(inventory, D[t][agent+1], S[t+1][agent]))

            # for last agent in supply chain: update state t+1 with inventory +␣
↪order of t-1 (delivers from warehouse)
            else:
                S[t+1][agent] = inventory + O[t-1][agent]
                print('Inventory {} receiving order +{} of t-1: {}'.
↪format(inventory, O[t-1][agent], S[t+1][agent]))

            # update coded states
            CS[t+1][agent] = coded_state(S[t+1][agent])

            # calculate agent's costs (onhand inventory holding costs + penalty␣
↪costs)
            r[t][agent] = 1 * max(S[t+1][agent], 0) + 2 * (x[t][agent] -␣
↪D[t][agent]) #backlog
            print('Costs r: {}\n'.format(r[t][agent]))

        # calculate the total supply chain costs in t
        action = tuple(y[t])
        R[t] = np.sum(r[t])
        print('Supply Chain costs R in state {} with action {} at t={}: {}'.
↪format(state, action, t, R[t]))

        # increase time step t
        t += 1
        print('\n\n')

    # view last episode
    fun_episode(S, CS, D, O, x, y, r, head=False)


    # View results
    print(R)
```

```
    print(np.sum(R))

    return np.sum(R)
```

[ ]:
```
T = 35

# Set up Lists to store the results of each iteration/episode
S = [list(np.repeat(0, len(agents))) for i in range(T+1)]
CS = [list(np.repeat(0, len(agents))) for i in range(T+1)]
D = [list(np.repeat(0, len(agents))) for i in range(T+1)]
O = [list(np.repeat(0, len(agents))) for i in range(T+1)]
x = [list(np.repeat(0, len(agents))) for i in range(T+1)]
y = [list(np.repeat(0, len(agents))) for i in range(T+1)]
r = [list(np.repeat(0, len(agents))) for i in range(T+1)]
R = [0 for i in range(T)]

# Extract optimal policy of Q-values
opt_policy = df_Q_values.idxmin(axis=1)
optimum_policy =[]
benchmark1 = []
benchmark2 = []
```

[ ]:
```
# view optimum policy
display(opt_policy)
```

```
(1, 1, 1)    (0, 3, 0)
(1, 1, 2)    (3, 1, 3)
(1, 1, 3)    (3, 1, 2)
(1, 1, 4)    (3, 1, 0)
(1, 1, 5)    (2, 3, 3)
               …
(5, 5, 1)    (0, 0, 0)
(5, 5, 2)    (0, 1, 3)
(5, 5, 3)    (2, 1, 0)
(5, 5, 4)    (2, 0, 2)
(5, 5, 5)    (2, 3, 1)
Length: 125, dtype: object
```

[ ]:
```
# view main problem set output
main_function(problem_set=main_set, policy='optimum', t=0, T=35)
```

```
-----------------time step t=0----------------
State S: (10, 10, 10) and coded state CS: (4, 4, 4)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1: 15
Distribution size D: 10
Backlog size: 5
Action y: 1
```

```
Ordering size O: 16

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 0
Distribution size D: 0
Backlog size: 0
Action y: 3
Ordering size O: 3

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 0
Distribution size D: 0
Backlog size: 0
Action y: 0
Ordering size O: 0

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -5 after receiving order +0 of t: -5
Costs r: 10

------------------Distributor------------------
Inventory 10 after receiving order +0 of t: 10
Costs r: 10

------------------Manufacturer------------------
Inventory 10 receiving order +0 of t-1: 10
Costs r: 10

Supply Chain costs R in state (10, 10, 10) with action (1, 3, 0) at t=0: 30



------------------time step t=1----------------
State S: (-5, 10, 10) and coded state CS: (2, 4, 4)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
10 + 5 = 15
Distribution size D (delivery of upstream agent in t-1): 0
Backlog size: 15
Action y: 2
Ordering size O: 12

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 16
Distribution size D: 10
Backlog size: 6
Action y: 0
```

```
Ordering size O: 16

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 3
Distribution size D: 3
Backlog size: 0
Action y: 1
Ordering size O: 4

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -15 after receiving order +10 of t: -5
Costs r: 30

------------------Distributor------------------
Inventory -6 after receiving order +3 of t: -3
Costs r: 12

------------------Manufacturer------------------
Inventory 7 receiving order +0 of t-1: 7
Costs r: 7

Supply Chain costs R in state (-5, 10, 10) with action (2, 0, 1) at t=1: 49



-----------------time step t=2----------------
State S: (-5, -3, 7) and coded state CS: (2, 2, 4)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
8 + 5 = 13
Distribution size D (delivery of upstream agent in t-1): 10
Backlog size: 3
Action y: 0
Ordering size O: 8

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1 + demand of previous time steps:
12 + 3 = 15
Distribution size D (delivery of upstream agent in t-1): 3
Backlog size: 12
Action y: 3
Ordering size O: 15

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 16
Distribution size D: 7
Backlog size: 9
```

```
Action y: 2
Ordering size O: 18

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -13 after receiving order +3 of t: -10
Costs r: 6

------------------Distributor------------------
Inventory -15 after receiving order +7 of t: -8
Costs r: 24

------------------Manufacturer------------------
Inventory -9 receiving order +4 of t-1: -5
Costs r: 18

Supply Chain costs R in state (-5, -3, 7) with action (0, 3, 2) at t=2: 48



-----------------time step t=3----------------
State S: (-10, -8, -5) and coded state CS: (1, 1, 2)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
14 + 10 = 24
Distribution size D (delivery of upstream agent in t-1): 3
Backlog size: 21
Action y: 3
Ordering size O: 17

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1 + demand of previous time steps:
8 + 8 = 16
Distribution size D (delivery of upstream agent in t-1): 7
Backlog size: 9
Action y: 1
Ordering size O: 9

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1 + demand of previous time steps:
15 + 5 = 20
Distribution size D (delivery of warehouse): 18
Backlog size: 2
Action y: 3
Ordering size O: 18

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
```

```
Inventory -24 after receiving order +7 of t: -17
Costs r: 42

------------------Distributor------------------
Inventory -16 after receiving order +18 of t: 2
Costs r: 20

------------------Manufacturer------------------
Inventory -20 receiving order +18 of t-1: -2
Costs r: 4

Supply Chain costs R in state (-10, -8, -5) with action (3, 1, 3) at t=3: 66



-----------------time step t=4----------------
State S: (-17, 2, -2) and coded state CS: (1, 3, 2)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
9 + 17 = 26
Distribution size D (delivery of upstream agent in t-1): 7
Backlog size: 19
Action y: 1
Ordering size O: 10

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 17
Distribution size D: 17
Backlog size: 0
Action y: 1
Ordering size O: 18

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1 + demand of previous time steps:
9 + 2 = 11
Distribution size D (delivery of warehouse): 11
Backlog size: 0
Action y: 3
Ordering size O: 12

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -26 after receiving order +17 of t: -9
Costs r: 38

------------------Distributor------------------
Inventory -15 after receiving order +11 of t: -4
Costs r: 0
```

```
------------------Manufacturer------------------
Inventory -11 receiving order +18 of t-1: 7
Costs r: 7

Supply Chain costs R in state (-17, 2, -2) with action (1, 1, 3) at t=4: 45



-----------------time step t=5----------------
State S: (-9, -4, 7) and coded state CS: (1, 2, 4)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
3 + 9 = 12
Distribution size D (delivery of upstream agent in t-1): 12
Backlog size: 0
Action y: 3
Ordering size O: 6

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1 + demand of previous time steps:
10 + 4 = 14
Distribution size D (delivery of upstream agent in t-1): 11
Backlog size: 3
Action y: 1
Ordering size O: 11

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 18
Distribution size D: 12
Backlog size: 6
Action y: 1
Ordering size O: 19

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -12 after receiving order +11 of t: -1
Costs r: 0

------------------Distributor------------------
Inventory -14 after receiving order +12 of t: -2
Costs r: 6

------------------Manufacturer------------------
Inventory -11 receiving order +12 of t-1: 1
Costs r: 13

Supply Chain costs R in state (-9, -4, 7) with action (3, 1, 1) at t=5: 19
```

```
-----------------time step t=6----------------
State S: (-1, -2, 1) and coded state CS: (2, 2, 3)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
13 + 1 = 14
Distribution size D (delivery of upstream agent in t-1): 11
Backlog size: 3
Action y: 1
Ordering size O: 14

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1 + demand of previous time steps:
6 + 2 = 8
Distribution size D (delivery of upstream agent in t-1): 8
Backlog size: 0
Action y: 0
Ordering size O: 6

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 11
Distribution size D: 11
Backlog size: 0
Action y: 3
Ordering size O: 14

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -14 after receiving order +8 of t: -6
Costs r: 6

------------------Distributor------------------
Inventory -8 after receiving order +11 of t: 3
Costs r: 3

------------------Manufacturer------------------
Inventory -10 receiving order +19 of t-1: 9
Costs r: 9

Supply Chain costs R in state (-1, -2, 1) with action (1, 0, 3) at t=6: 18



-----------------time step t=7----------------
State S: (-6, 3, 9) and coded state CS: (1, 3, 4)
------------------Retailer------------------
```

```
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
2 + 6 = 8
Distribution size D (delivery of upstream agent in t-1): 8
Backlog size: 0
Action y: 2
Ordering size O: 4

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 14
Distribution size D: 11
Backlog size: 3
Action y: 1
Ordering size O: 15

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 6
Distribution size D: 6
Backlog size: 0
Action y: 0
Ordering size O: 6

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -8 after receiving order +11 of t: 3
Costs r: 3

------------------Distributor------------------
Inventory -11 after receiving order +6 of t: -5
Costs r: 6

------------------Manufacturer------------------
Inventory 3 receiving order +14 of t-1: 17
Costs r: 17

Supply Chain costs R in state (-6, 3, 9) with action (2, 1, 0) at t=7: 26



------------------time step t=8----------------
State S: (3, -5, 17) and coded state CS: (3, 2, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1: 13
Distribution size D: 11
Backlog size: 2
Action y: 3
Ordering size O: 16

------------------Distributor------------------
```

```
Demand x from downstream (Retailer) at t-1 + demand of previous time steps:
4 + 5 = 9
Distribution size D (delivery of upstream agent in t-1): 6
Backlog size: 3
Action y: 3
Ordering size O: 7

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 15
Distribution size D: 15
Backlog size: 0
Action y: 0
Ordering size O: 15

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -10 after receiving order +6 of t: -4
Costs r: 4

------------------Distributor------------------
Inventory -9 after receiving order +15 of t: 6
Costs r: 12

------------------Manufacturer------------------
Inventory 2 receiving order +6 of t-1: 8
Costs r: 8

Supply Chain costs R in state (3, -5, 17) with action (3, 3, 0) at t=8: 24



-----------------time step t=9----------------
State S: (-4, 6, 8) and coded state CS: (2, 4, 4)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
11 + 4 = 15
Distribution size D (delivery of upstream agent in t-1): 6
Backlog size: 9
Action y: 2
Ordering size O: 13

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 16
Distribution size D: 15
Backlog size: 1
Action y: 0
Ordering size O: 16
```

```
------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 7
Distribution size D: 7
Backlog size: 0
Action y: 1
Ordering size O: 8

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -15 after receiving order +15 of t: 0
Costs r: 18

------------------Distributor------------------
Inventory -10 after receiving order +7 of t: -3
Costs r: 2

------------------Manufacturer------------------
Inventory 1 receiving order +15 of t-1: 16
Costs r: 16

Supply Chain costs R in state (-4, 6, 8) with action (2, 0, 1) at t=9: 36



-----------------time step t=10----------------
State S: (0, -3, 16) and coded state CS: (3, 2, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1: 3
Distribution size D: 3
Backlog size: 0
Action y: 3
Ordering size O: 6

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1 + demand of previous time steps:
13 + 3 = 16
Distribution size D (delivery of upstream agent in t-1): 7
Backlog size: 9
Action y: 3
Ordering size O: 16

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 16
Distribution size D: 16
Backlog size: 0
Action y: 0
Ordering size O: 16
```

```
UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -3 after receiving order +7 of t: 4
Costs r: 4

------------------Distributor------------------
Inventory -16 after receiving order +16 of t: 0
Costs r: 18

------------------Manufacturer------------------
Inventory 0 receiving order +8 of t-1: 8
Costs r: 8

Supply Chain costs R in state (0, -3, 16) with action (3, 3, 0) at t=10: 30



-----------------time step t=11----------------
State S: (4, 0, 8) and coded state CS: (3, 3, 4)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1: 4
Distribution size D: 4
Backlog size: 0
Action y: 1
Ordering size O: 5

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 6
Distribution size D: 6
Backlog size: 0
Action y: 1
Ordering size O: 7

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 16
Distribution size D: 16
Backlog size: 0
Action y: 0
Ordering size O: 16

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory 0 after receiving order +6 of t: 6
Costs r: 6

------------------Distributor------------------
Inventory -6 after receiving order +16 of t: 10
Costs r: 10
```

```
------------------Manufacturer------------------
Inventory -8 receiving order +16 of t-1: 8
Costs r: 8

Supply Chain costs R in state (4, 0, 8) with action (1, 1, 0) at t=11: 24



-----------------time step t=12----------------
State S: (6, 10, 8) and coded state CS: (4, 4, 4)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1: 6
Distribution size D: 6
Backlog size: 0
Action y: 1
Ordering size O: 7

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 5
Distribution size D: 5
Backlog size: 0
Action y: 3
Ordering size O: 8

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 7
Distribution size D: 7
Backlog size: 0
Action y: 0
Ordering size O: 7

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory 0 after receiving order +5 of t: 5
Costs r: 5

------------------Distributor------------------
Inventory 5 after receiving order +7 of t: 12
Costs r: 12

------------------Manufacturer------------------
Inventory 1 receiving order +16 of t-1: 17
Costs r: 17

Supply Chain costs R in state (6, 10, 8) with action (1, 3, 0) at t=12: 34
```

```
-----------------time step t=13----------------
State S: (5, 12, 17) and coded state CS: (4, 5, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1: 11
Distribution size D: 5
Backlog size: 6
Action y: 2
Ordering size O: 13

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 7
Distribution size D: 7
Backlog size: 0
Action y: 0
Ordering size O: 7

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 8
Distribution size D: 8
Backlog size: 0
Action y: 0
Ordering size O: 8

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -6 after receiving order +7 of t: 1
Costs r: 13

------------------Distributor------------------
Inventory 5 after receiving order +8 of t: 13
Costs r: 13

------------------Manufacturer------------------
Inventory 9 receiving order +7 of t-1: 16
Costs r: 16

Supply Chain costs R in state (5, 12, 17) with action (2, 0, 0) at t=13: 42



-----------------time step t=14----------------
State S: (1, 13, 16) and coded state CS: (3, 5, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1: 15
Distribution size D: 7
Backlog size: 8
Action y: 2
```

```
Ordering size O: 17

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 13
Distribution size D: 13
Backlog size: 0
Action y: 0
Ordering size O: 13

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 7
Distribution size D: 7
Backlog size: 0
Action y: 3
Ordering size O: 10

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -14 after receiving order +13 of t: -1
Costs r: 16

------------------Distributor------------------
Inventory 0 after receiving order +7 of t: 7
Costs r: 7

------------------Manufacturer------------------
Inventory 9 receiving order +8 of t-1: 17
Costs r: 17

Supply Chain costs R in state (1, 13, 16) with action (2, 0, 3) at t=14: 40



------------------time step t=15----------------
State S: (-1, 7, 17) and coded state CS: (2, 4, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
12 + 1 = 13
Distribution size D (delivery of upstream agent in t-1): 13
Backlog size: 0
Action y: 2
Ordering size O: 14

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 17
Distribution size D: 7
Backlog size: 10
Action y: 0
```

```
Ordering size O: 17

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 13
Distribution size D: 13
Backlog size: 0
Action y: 0
Ordering size O: 13

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -13 after receiving order +7 of t: -6
Costs r: 0

------------------Distributor------------------
Inventory -10 after receiving order +13 of t: 3
Costs r: 23

------------------Manufacturer------------------
Inventory 4 receiving order +10 of t-1: 14
Costs r: 14

Supply Chain costs R in state (-1, 7, 17) with action (2, 0, 0) at t=15: 37



-----------------time step t=16----------------
State S: (-6, 3, 14) and coded state CS: (1, 3, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
15 + 6 = 21
Distribution size D (delivery of upstream agent in t-1): 7
Backlog size: 14
Action y: 2
Ordering size O: 17

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 14
Distribution size D: 13
Backlog size: 1
Action y: 1
Ordering size O: 15

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 17
Distribution size D: 14
Backlog size: 3
Action y: 2
```

```
Ordering size O: 19

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -21 after receiving order +13 of t: -8
Costs r: 28

------------------Distributor------------------
Inventory -11 after receiving order +14 of t: 3
Costs r: 5

------------------Manufacturer------------------
Inventory -3 receiving order +13 of t-1: 10
Costs r: 16

Supply Chain costs R in state (-6, 3, 14) with action (2, 1, 2) at t=16: 49



-----------------time step t=17----------------
State S: (-8, 3, 10) and coded state CS: (1, 3, 4)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
4 + 8 = 12
Distribution size D (delivery of upstream agent in t-1): 12
Backlog size: 0
Action y: 2
Ordering size O: 6

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 17
Distribution size D: 14
Backlog size: 3
Action y: 1
Ordering size O: 18

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 15
Distribution size D: 15
Backlog size: 0
Action y: 0
Ordering size O: 15

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -12 after receiving order +14 of t: 2
Costs r: 2
```

```
------------------Distributor------------------
Inventory -14 after receiving order +15 of t: 1
Costs r: 7

------------------Manufacturer------------------
Inventory -5 receiving order +19 of t-1: 14
Costs r: 14

Supply Chain costs R in state (-8, 3, 10) with action (2, 1, 0) at t=17: 23



-----------------time step t=18----------------
State S: (2, 1, 14) and coded state CS: (3, 3, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1: 12
Distribution size D: 12
Backlog size: 0
Action y: 0
Ordering size O: 12

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 6
Distribution size D: 6
Backlog size: 0
Action y: 0
Ordering size O: 6

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 18
Distribution size D: 15
Backlog size: 3
Action y: 0
Ordering size O: 18

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -10 after receiving order +6 of t: -4
Costs r: 0

------------------Distributor------------------
Inventory -5 after receiving order +15 of t: 10
Costs r: 10

------------------Manufacturer------------------
Inventory -4 receiving order +15 of t-1: 11
Costs r: 17
```

```
Supply Chain costs R in state (2, 1, 14) with action (0, 0, 0) at t=18: 27



-----------------time step t=19---------------
State S: (-4, 10, 11) and coded state CS: (2, 4, 4)
-----------------Retailer-----------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
3 + 4 = 7
Distribution size D (delivery of upstream agent in t-1): 6
Backlog size: 1
Action y: 2
Ordering size O: 5

-----------------Distributor-----------------
Demand x from downstream (Retailer) at t-1: 12
Distribution size D: 12
Backlog size: 0
Action y: 0
Ordering size O: 12

-----------------Manufacturer-----------------
Demand x from downstream (Distributor) at t-1: 6
Distribution size D: 6
Backlog size: 0
Action y: 1
Ordering size O: 7

UPDATING STATES AND CALCULATING COSTS PER AGENT
-----------------Retailer-----------------
Inventory -7 after receiving order +12 of t: 5
Costs r: 7

-----------------Distributor-----------------
Inventory -2 after receiving order +6 of t: 4
Costs r: 4

-----------------Manufacturer-----------------
Inventory 5 receiving order +18 of t-1: 23
Costs r: 23

Supply Chain costs R in state (-4, 10, 11) with action (2, 0, 1) at t=19: 34



-----------------time step t=20---------------
State S: (5, 4, 23) and coded state CS: (4, 3, 5)
-----------------Retailer-----------------
```

```
Demand x from downstream (Customer) at t-1: 13
Distribution size D: 12
Backlog size: 1
Action y: 2
Ordering size O: 15

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 5
Distribution size D: 5
Backlog size: 0
Action y: 1
Ordering size O: 6

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 12
Distribution size D: 12
Backlog size: 0
Action y: 2
Ordering size O: 14

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -8 after receiving order +5 of t: -3
Costs r: 2

------------------Distributor------------------
Inventory -1 after receiving order +12 of t: 11
Costs r: 11

------------------Manufacturer------------------
Inventory 11 receiving order +7 of t-1: 18
Costs r: 18

Supply Chain costs R in state (5, 4, 23) with action (2, 1, 2) at t=20: 31



-----------------time step t=21----------------
State S: (-3, 11, 18) and coded state CS: (2, 4, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
10 + 3 = 13
Distribution size D (delivery of upstream agent in t-1): 5
Backlog size: 8
Action y: 2
Ordering size O: 12

------------------Distributor------------------
```

```
Demand x from downstream (Retailer) at t-1: 15
Distribution size D: 12
Backlog size: 3
Action y: 0
Ordering size O: 15

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 6
Distribution size D: 6
Backlog size: 0
Action y: 0
Ordering size O: 6

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -13 after receiving order +12 of t: -1
Costs r: 16

------------------Distributor------------------
Inventory -4 after receiving order +6 of t: 2
Costs r: 8

------------------Manufacturer------------------
Inventory 12 receiving order +14 of t-1: 26
Costs r: 26

Supply Chain costs R in state (-3, 11, 18) with action (2, 0, 0) at t=21: 50



------------------time step t=22----------------
State S: (-1, 2, 26) and coded state CS: (2, 3, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
15 + 1 = 16
Distribution size D (delivery of upstream agent in t-1): 12
Backlog size: 4
Action y: 3
Ordering size O: 18

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 12
Distribution size D: 6
Backlog size: 6
Action y: 2
Ordering size O: 14

------------------Manufacturer------------------
```

```
Demand x from downstream (Distributor) at t-1: 15
Distribution size D: 15
Backlog size: 0
Action y: 3
Ordering size O: 18

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -16 after receiving order +6 of t: -10
Costs r: 8

------------------Distributor------------------
Inventory -10 after receiving order +15 of t: 5
Costs r: 17

------------------Manufacturer------------------
Inventory 11 receiving order +6 of t-1: 17
Costs r: 17

Supply Chain costs R in state (-1, 2, 26) with action (3, 2, 3) at t=22: 42



-----------------time step t=23----------------
State S: (-10, 5, 17) and coded state CS: (1, 4, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
15 + 10 = 25
Distribution size D (delivery of upstream agent in t-1): 6
Backlog size: 19
Action y: 2
Ordering size O: 17

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 18
Distribution size D: 15
Backlog size: 3
Action y: 1
Ordering size O: 19

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 14
Distribution size D: 14
Backlog size: 0
Action y: 2
Ordering size O: 16

UPDATING STATES AND CALCULATING COSTS PER AGENT
```

```
------------------Retailer------------------
Inventory -25 after receiving order +15 of t: -10
Costs r: 38

------------------Distributor------------------
Inventory -13 after receiving order +14 of t: 1
Costs r: 7

------------------Manufacturer------------------
Inventory 3 receiving order +18 of t-1: 21
Costs r: 21

Supply Chain costs R in state (-10, 5, 17) with action (2, 1, 2) at t=23: 66



-----------------time step t=24----------------
State S: (-10, 1, 21) and coded state CS: (1, 3, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
3 + 10 = 13
Distribution size D (delivery of upstream agent in t-1): 13
Backlog size: 0
Action y: 2
Ordering size O: 5

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 17
Distribution size D: 14
Backlog size: 3
Action y: 1
Ordering size O: 18

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 19
Distribution size D: 19
Backlog size: 0
Action y: 2
Ordering size O: 21

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -13 after receiving order +14 of t: 1
Costs r: 1

------------------Distributor------------------
Inventory -16 after receiving order +19 of t: 3
Costs r: 9
```

```
------------------Manufacturer------------------
Inventory 2 receiving order +16 of t-1: 18
Costs r: 18

Supply Chain costs R in state (-10, 1, 21) with action (2, 1, 2) at t=24: 28



-----------------time step t=25----------------
State S: (1, 3, 18) and coded state CS: (3, 3, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1: 11
Distribution size D: 11
Backlog size: 0
Action y: 0
Ordering size O: 11

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 5
Distribution size D: 5
Backlog size: 0
Action y: 0
Ordering size O: 5

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 18
Distribution size D: 18
Backlog size: 0
Action y: 0
Ordering size O: 18

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -10 after receiving order +5 of t: -5
Costs r: 0

------------------Distributor------------------
Inventory -2 after receiving order +18 of t: 16
Costs r: 16

------------------Manufacturer------------------
Inventory 0 receiving order +21 of t-1: 21
Costs r: 21

Supply Chain costs R in state (1, 3, 18) with action (0, 0, 0) at t=25: 37
```

```
-----------------time step t=26----------------
State S: (-5, 16, 21) and coded state CS: (2, 5, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
1 + 5 = 6
Distribution size D (delivery of upstream agent in t-1): 5
Backlog size: 1
Action y: 0
Ordering size O: 1

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 11
Distribution size D: 11
Backlog size: 0
Action y: 0
Ordering size O: 11

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 5
Distribution size D: 5
Backlog size: 0
Action y: 1
Ordering size O: 6

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -6 after receiving order +11 of t: 5
Costs r: 7

------------------Distributor------------------
Inventory 5 after receiving order +5 of t: 10
Costs r: 10

------------------Manufacturer------------------
Inventory 16 receiving order +18 of t-1: 34
Costs r: 34

Supply Chain costs R in state (-5, 16, 21) with action (0, 0, 1) at t=26: 51



-----------------time step t=27----------------
State S: (5, 10, 34) and coded state CS: (4, 4, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1: 13
Distribution size D: 11
Backlog size: 2
```

```
Action y: 3
Ordering size O: 16

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 1
Distribution size D: 1
Backlog size: 0
Action y: 0
Ordering size O: 1

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 11
Distribution size D: 11
Backlog size: 0
Action y: 3
Ordering size O: 14

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -8 after receiving order +1 of t: -7
Costs r: 4

------------------Distributor------------------
Inventory 9 after receiving order +11 of t: 20
Costs r: 20

------------------Manufacturer------------------
Inventory 23 receiving order +6 of t-1: 29
Costs r: 29

Supply Chain costs R in state (5, 10, 34) with action (3, 0, 3) at t=27: 53



-----------------time step t=28----------------
State S: (-7, 20, 29) and coded state CS: (1, 5, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
10 + 7 = 17
Distribution size D (delivery of upstream agent in t-1): 1
Backlog size: 16
Action y: 3
Ordering size O: 13

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 16
Distribution size D: 16
Backlog size: 0
```

```
Action y: 2
Ordering size O: 18

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 1
Distribution size D: 1
Backlog size: 0
Action y: 1
Ordering size O: 2

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -17 after receiving order +16 of t: -1
Costs r: 32

------------------Distributor------------------
Inventory 4 after receiving order +1 of t: 5
Costs r: 5

------------------Manufacturer------------------
Inventory 28 receiving order +14 of t-1: 42
Costs r: 42

Supply Chain costs R in state (-7, 20, 29) with action (3, 2, 1) at t=28: 79



------------------time step t=29----------------
State S: (-1, 5, 42) and coded state CS: (2, 4, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
10 + 1 = 11
Distribution size D (delivery of upstream agent in t-1): 11
Backlog size: 0
Action y: 2
Ordering size O: 12

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 13
Distribution size D: 5
Backlog size: 8
Action y: 0
Ordering size O: 13

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 18
Distribution size D: 18
Backlog size: 0
```

```
Action y: 0
Ordering size O: 18

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -11 after receiving order +5 of t: -6
Costs r: 0

------------------Distributor------------------
Inventory -8 after receiving order +18 of t: 10
Costs r: 26

------------------Manufacturer------------------
Inventory 24 receiving order +2 of t-1: 26
Costs r: 26

Supply Chain costs R in state (-1, 5, 42) with action (2, 0, 0) at t=29: 52



-----------------time step t=30----------------
State S: (-6, 10, 26) and coded state CS: (1, 4, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1 + demand of previous time steps:
0 + 6 = 6
Distribution size D (delivery of upstream agent in t-1): 5
Backlog size: 1
Action y: 2
Ordering size O: 2

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 12
Distribution size D: 12
Backlog size: 0
Action y: 1
Ordering size O: 13

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 13
Distribution size D: 13
Backlog size: 0
Action y: 2
Ordering size O: 15

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -6 after receiving order +12 of t: 6
Costs r: 8
```

```
------------------Distributor------------------
Inventory -2 after receiving order +13 of t: 11
Costs r: 11

------------------Manufacturer------------------
Inventory 13 receiving order +18 of t-1: 31
Costs r: 31

Supply Chain costs R in state (-6, 10, 26) with action (2, 1, 2) at t=30: 50



-----------------time step t=31----------------
State S: (6, 11, 31) and coded state CS: (4, 4, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1: 0
Distribution size D: 0
Backlog size: 0
Action y: 3
Ordering size O: 3

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 2
Distribution size D: 2
Backlog size: 0
Action y: 0
Ordering size O: 2

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 13
Distribution size D: 13
Backlog size: 0
Action y: 3
Ordering size O: 16

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory 6 after receiving order +2 of t: 8
Costs r: 8

------------------Distributor------------------
Inventory 9 after receiving order +13 of t: 22
Costs r: 22

------------------Manufacturer------------------
Inventory 18 receiving order +15 of t-1: 33
Costs r: 33
```

```
Supply Chain costs R in state (6, 11, 31) with action (3, 0, 3) at t=31: 63



-----------------time step t=32----------------
State S: (8, 22, 33) and coded state CS: (4, 5, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1: 8
Distribution size D: 8
Backlog size: 0
Action y: 2
Ordering size O: 10

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 3
Distribution size D: 3
Backlog size: 0
Action y: 0
Ordering size O: 3

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 2
Distribution size D: 2
Backlog size: 0
Action y: 0
Ordering size O: 2

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory 0 after receiving order +3 of t: 3
Costs r: 3

------------------Distributor------------------
Inventory 19 after receiving order +2 of t: 21
Costs r: 21

------------------Manufacturer------------------
Inventory 31 receiving order +16 of t-1: 47
Costs r: 47

Supply Chain costs R in state (8, 22, 33) with action (2, 0, 0) at t=32: 71



-----------------time step t=33----------------
State S: (3, 21, 47) and coded state CS: (3, 5, 5)
------------------Retailer------------------
```

```
Demand x from downstream (Customer) at t-1: 0
Distribution size D: 0
Backlog size: 0
Action y: 2
Ordering size O: 2

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 10
Distribution size D: 10
Backlog size: 0
Action y: 0
Ordering size O: 10

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 3
Distribution size D: 3
Backlog size: 0
Action y: 3
Ordering size O: 6

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory 3 after receiving order +10 of t: 13
Costs r: 13

------------------Distributor------------------
Inventory 11 after receiving order +3 of t: 14
Costs r: 14

------------------Manufacturer------------------
Inventory 44 receiving order +2 of t-1: 46
Costs r: 46

Supply Chain costs R in state (3, 21, 47) with action (2, 0, 3) at t=33: 73



-----------------time step t=34----------------
State S: (13, 14, 46) and coded state CS: (5, 5, 5)
------------------Retailer------------------
Demand x from downstream (Customer) at t-1: 14
Distribution size D: 13
Backlog size: 1
Action y: 2
Ordering size O: 16

------------------Distributor------------------
Demand x from downstream (Retailer) at t-1: 2
```

```
Distribution size D: 2
Backlog size: 0
Action y: 3
Ordering size O: 5

------------------Manufacturer------------------
Demand x from downstream (Distributor) at t-1: 10
Distribution size D: 10
Backlog size: 0
Action y: 1
Ordering size O: 11

UPDATING STATES AND CALCULATING COSTS PER AGENT
------------------Retailer------------------
Inventory -1 after receiving order +2 of t: 1
Costs r: 3

------------------Distributor------------------
Inventory 12 after receiving order +10 of t: 22
Costs r: 22

------------------Manufacturer------------------
Inventory 36 receiving order +6 of t-1: 42
Costs r: 42

Supply Chain costs R in state (13, 14, 46) with action (2, 3, 1) at t=34: 67
```

Time	Inventory/States S	Coded states CS	Demand x	Distribution amount D \
t=0	(10, 10, 10)	(4, 4, 4)	(15, 0, 0)	(10, 0, 0)
t=1	(-5, 10, 10)	(2, 4, 4)	(15, 16, 3)	(0, 10, 3)
t=2	(-5, -3, 7)	(2, 2, 4)	(13, 15, 16)	(10, 3, 7)
t=3	(-10, -8, -5)	(1, 1, 2)	(24, 16, 20)	(3, 7, 18)
t=4	(-17, 2, -2)	(1, 3, 2)	(26, 17, 11)	(7, 17, 11)
t=5	(-9, -4, 7)	(1, 2, 4)	(12, 14, 18)	(12, 11, 12)
t=6	(-1, -2, 1)	(2, 2, 3)	(14, 8, 11)	(11, 8, 11)
t=7	(-6, 3, 9)	(1, 3, 4)	(8, 14, 6)	(8, 11, 6)
t=8	(3, -5, 17)	(3, 2, 5)	(13, 9, 15)	(11, 6, 15)
t=9	(-4, 6, 8)	(2, 4, 4)	(15, 16, 7)	(6, 15, 7)
t=10	(0, -3, 16)	(3, 2, 5)	(3, 16, 16)	(3, 7, 16)
t=11	(4, 0, 8)	(3, 3, 4)	(4, 6, 16)	(4, 6, 16)
t=12	(6, 10, 8)	(4, 4, 4)	(6, 5, 7)	(6, 5, 7)
t=13	(5, 12, 17)	(4, 5, 5)	(11, 7, 8)	(5, 7, 8)
t=14	(1, 13, 16)	(3, 5, 5)	(15, 13, 7)	(7, 13, 7)
t=15	(-1, 7, 17)	(2, 4, 5)	(13, 17, 13)	(13, 7, 13)

t=16	(-6, 3, 14)	(1, 3, 5)	(21, 14, 17)	(7, 13, 14)
t=17	(-8, 3, 10)	(1, 3, 4)	(12, 17, 15)	(12, 14, 15)
t=18	(2, 1, 14)	(3, 3, 5)	(12, 6, 18)	(12, 6, 15)
t=19	(-4, 10, 11)	(2, 4, 4)	(7, 12, 6)	(6, 12, 6)
t=20	(5, 4, 23)	(4, 3, 5)	(13, 5, 12)	(12, 5, 12)
t=21	(-3, 11, 18)	(2, 4, 5)	(13, 15, 6)	(5, 12, 6)
t=22	(-1, 2, 26)	(2, 3, 5)	(16, 12, 15)	(12, 6, 15)
t=23	(-10, 5, 17)	(1, 4, 5)	(25, 18, 14)	(6, 15, 14)
t=24	(-10, 1, 21)	(1, 3, 5)	(13, 17, 19)	(13, 14, 19)
t=25	(1, 3, 18)	(3, 3, 5)	(11, 5, 18)	(11, 5, 18)
t=26	(-5, 16, 21)	(2, 5, 5)	(6, 11, 5)	(5, 11, 5)
t=27	(5, 10, 34)	(4, 4, 5)	(13, 1, 11)	(11, 1, 11)
t=28	(-7, 20, 29)	(1, 5, 5)	(17, 16, 1)	(1, 16, 1)
t=29	(-1, 5, 42)	(2, 4, 5)	(11, 13, 18)	(11, 5, 18)
t=30	(-6, 10, 26)	(1, 4, 5)	(6, 12, 13)	(5, 12, 13)
t=31	(6, 11, 31)	(4, 4, 5)	(0, 2, 13)	(0, 2, 13)
t=32	(8, 22, 33)	(4, 5, 5)	(8, 3, 2)	(8, 3, 2)
t=33	(3, 21, 47)	(3, 5, 5)	(0, 10, 3)	(0, 10, 3)
t=34	(13, 14, 46)	(5, 5, 5)	(14, 2, 10)	(13, 2, 10)
t=35	(1, 22, 42)	(3, 5, 5)	(0, 0, 0)	(0, 0, 0)

Time	Action y	Ordering size	Costs r
t=0	(1, 3, 0)	(16, 3, 0)	(10, 10, 10)
t=1	(2, 0, 1)	(12, 16, 4)	(30, 12, 7)
t=2	(0, 3, 2)	(8, 15, 18)	(6, 24, 18)
t=3	(3, 1, 3)	(17, 9, 18)	(42, 20, 4)
t=4	(1, 1, 3)	(10, 18, 12)	(38, 0, 7)
t=5	(3, 1, 1)	(6, 11, 19)	(0, 6, 13)
t=6	(1, 0, 3)	(14, 6, 14)	(6, 3, 9)
t=7	(2, 1, 0)	(4, 15, 6)	(3, 6, 17)
t=8	(3, 3, 0)	(16, 7, 15)	(4, 12, 8)
t=9	(2, 0, 1)	(13, 16, 8)	(18, 2, 16)
t=10	(3, 3, 0)	(6, 16, 16)	(4, 18, 8)
t=11	(1, 1, 0)	(5, 7, 16)	(6, 10, 8)
t=12	(1, 3, 0)	(7, 8, 7)	(5, 12, 17)
t=13	(2, 0, 0)	(13, 7, 8)	(13, 13, 16)
t=14	(2, 0, 3)	(17, 13, 10)	(16, 7, 17)
t=15	(2, 0, 0)	(14, 17, 13)	(0, 23, 14)
t=16	(2, 1, 2)	(17, 15, 19)	(28, 5, 16)
t=17	(2, 1, 0)	(6, 18, 15)	(2, 7, 14)
t=18	(0, 0, 0)	(12, 6, 18)	(0, 10, 17)
t=19	(2, 0, 1)	(5, 12, 7)	(7, 4, 23)
t=20	(2, 1, 2)	(15, 6, 14)	(2, 11, 18)
t=21	(2, 0, 0)	(12, 15, 6)	(16, 8, 26)
t=22	(3, 2, 3)	(18, 14, 18)	(8, 17, 17)
t=23	(2, 1, 2)	(17, 19, 16)	(38, 7, 21)
t=24	(2, 1, 2)	(5, 18, 21)	(1, 9, 18)

```
t=25  (0, 0, 0)   (11, 5, 18)   (0, 16, 21)
t=26  (0, 0, 1)    (1, 11, 6)   (7, 10, 34)
t=27  (3, 0, 3)   (16, 1, 14)   (4, 20, 29)
t=28  (3, 2, 1)   (13, 18, 2)   (32, 5, 42)
t=29  (2, 0, 0)  (12, 13, 18)   (0, 26, 26)
t=30  (2, 1, 2)   (2, 13, 15)   (8, 11, 31)
t=31  (3, 0, 3)    (3, 2, 16)   (8, 22, 33)
t=32  (2, 0, 0)    (10, 3, 2)   (3, 21, 47)
t=33  (2, 0, 3)    (2, 10, 6)  (13, 14, 46)
t=34  (2, 3, 1)   (16, 5, 11)   (3, 22, 42)
t=35  (0, 0, 0)     (0, 0, 0)     (0, 0, 0)

[30, 49, 48, 66, 45, 19, 18, 26, 24, 36, 30, 24, 34, 42, 40, 37, 49, 23, 27, 34,
31, 50, 42, 66, 28, 37, 51, 53, 79, 52, 50, 63, 71, 73, 67]
1514
```

[ ]:
```
1514
```

[ ]:
```
%%capture

optimum_policy.append(main_function(problem_set=main_set, policy='optimum',␣
 ↪t=0, T=35))
optimum_policy.append(main_function(problem_set=set1, policy='optimum', t=0,␣
 ↪T=35))
optimum_policy.append(main_function(problem_set=set2, policy='optimum', t=0,␣
 ↪T=35))
optimum_policy.append(main_function(problem_set=set3, policy='optimum', t=0,␣
 ↪T=35))
```

1.0.4 Benchmark policy 1: Order = demand (y=0)

[ ]:
```
T = 35

# Set up Lists to store the results of each iteration/episode
S = [list(np.repeat(0, len(agents))) for i in range(T+1)]
CS = [list(np.repeat(0, len(agents))) for i in range(T+1)]
D = [list(np.repeat(0, len(agents))) for i in range(T+1)]
O = [list(np.repeat(0, len(agents))) for i in range(T+1)]
x = [list(np.repeat(0, len(agents))) for i in range(T+1)]
y = [list(np.repeat(0, len(agents))) for i in range(T+1)]
r = [list(np.repeat(0, len(agents))) for i in range(T+1)]
R = [0 for i in range(T)]
```

[ ]:
```
%%capture

benchmark1.append(main_function(problem_set=set1, policy='zero', t=0, T=35))
benchmark1.append(main_function(problem_set=set1, policy='zero', t=0, T=35))
benchmark1.append(main_function(problem_set=set2, policy='zero', t=0, T=35))
```

```
benchmark1.append(main_function(problem_set=set3, policy='zero', t=0, T=35))
```

1.0.5 Benchmark policy 2: Order = demand + random y [0, 3]

[ ]:
```
T = 35

# Set up Lists to store the results of each iteration/episode
S = [list(np.repeat(0, len(agents))) for i in range(T+1)]
CS = [list(np.repeat(0, len(agents))) for i in range(T+1)]
D = [list(np.repeat(0, len(agents))) for i in range(T+1)]
O = [list(np.repeat(0, len(agents))) for i in range(T+1)]
x = [list(np.repeat(0, len(agents))) for i in range(T+1)]
y = [list(np.repeat(0, len(agents))) for i in range(T+1)]
r = [list(np.repeat(0, len(agents))) for i in range(T+1)]
R = [0 for i in range(T)]
```

[ ]:
```
%%capture

benchmark2.append(main_function(problem_set=set1, policy='random', t=0, T=35))
benchmark2.append(main_function(problem_set=set1, policy='random', t=0, T=35))
benchmark2.append(main_function(problem_set=set2, policy='random', t=0, T=35))
benchmark2.append(main_function(problem_set=set3, policy='random', t=0, T=35))
```

1.0.6 Compare costs for all problem sets

[ ]:
```
print('Sum of costs per problem of optimum policy with Q-vales lernt by RLOM:
  ↪\n{}'.format(optimum_policy))
print('Mean costs: {}\n'.format(np.round(np.mean(optimum_policy))))

print('Sum of costs per problem of benchmark1 (y=0):\n{}'.format(benchmark1))
print('Mean costs: {}\n'.format(np.round(np.mean(benchmark1))))

print('Sum of costs per problem of benchmark2 (random):\n{}'.format(benchmark2))
print('Mean costs: {}\n'.format(np.round(np.mean(benchmark2))))
```

```
Sum of costs per problem of optimum policy with Q-vales lernt by RLOM:
[1514, 1853, 1787, 1970]
Mean costs: 1781.0

Sum of costs per problem of benchmark1 (y=0):
[2665, 2665, 2742, 2009]
Mean costs: 2520.0

Sum of costs per problem of benchmark2 (random):
[2645, 2313, 2133, 2073]
Mean costs: 2291.0
```