Shapley Coalitions for Prosumers in NRG-X-Change Network

Prosumer Synthesized Data from EIA.gov

The data gathered is from EIA.gov. The data was then used to synthesize typical prosumer consumption and generation over the course of 12 months. Volatitlity in the usage and generation was added as a normal distribution with a given variance to simulate real world conditions. The data is pulled from a local file and then sorted by timestamp. The fields are grouped by id and by time. Grouping by time allows for settlement calculations to occur at each time interval.

In [115…

```
import pandas as pd
import numpy as np
####################################################################
# Data Gathering
####################################################################
# Data gathering and synthesization has been done in a seperate module.
# import the external data gathering set. Pull data from a dataset
# of randomly insantiated prosumer, synthesized from EIA.gov data trends
#
import p0_data_gather as p0
# Set path of dataset file containing all parameterized values
data_set_path = 'data/prosumer_N10_all_20210305_1129.csv'
# Set initial conditions for prosumers dataset
dem_mean = 1100
gen_mean = 1300
# Set number of prosumers, an array of N[] values for multiple experiments
number_of_prosumers = [2,3,4,5,6,7,8,9,10]
# Trials [] holds session data for each N itteration of prosumers
trials = []
for N in number_of_prosumers:
    # Call the data as a query with the instantiated values
    prosumers_n = p0.get_data(path=data_set_path,query=f'id > 0 & id<={N} & demand_std == {dem_mean*0.20} & generation_s
    ####################################################################
    # Data Wrangling : Split data by time 't'
    ####################################################################
    # Wrangle data into monthly timesteps so that each time step
    # could be processed individually as a market payment for each prosumer
    prosumers_n_t = [pd.DataFrame(y) for x, y in prosumers_n.groupby('time', as_index=False)]
    trials.append({"N":N, "prosumers_n":prosumers_n, "prosumers_n_t":prosumers_n_t})
```

NRG-X-Change Methods

What is NRG-X-Change? "In this paper we propose NRG-X-Change — anovel mechanism for trading of locally produced re-newable energy that does not rely on an energy mar-ket or matching of orders ahead of time. In our modellocally produced energy is continuously fed into thegrid and payment is received based on actual usage,rather than predicted, as consumption is measured bythe DSO (Distribution Service Operator) and billed in near real-time." (Mihail Mihaylov)

Modeling a Prosumer Market

We will leverage an NRGX-Change based market of prosumers on an micro-grid. Any excess energy that is not consumed by the micro-grid is not considered at this time. The goal of each prosumer would be to offset the demand of the micro-grid. In most cases the total demand would need to be supplemented by the larger Grid at some retail electricity price. The network will use the payout and consumption functions for NRGX-Change as each prosumer generates every month.

The prosumers (P) are indicated on the diagram as generators of electricity. In this scenario solar PV is shown as a generation source. The Distribution Service Operator (DSO) is in charge of consuming excess Net Energy that is generated by the prosumer and it is not consumed by the prosumer. A consumer (C) is the term for a residence that has to consumer more energy that it can produce. It will need to pay for electricity and the DSO would charge it at some price. The NRG-X-Change function will charge the consumer and pay the prosumer for electricity consumed or generated.

prosumer_network_model

Prosumer Payment Function

The NRG-X-Change as described by the authors performs a dynamic payment to prosumers that are capable of meeting the demand of the micro-grid. The micro-grid is made of prosumers and consumers. As the load demand spikes the pricing for net generation also spikes to meet the demand. When there is too much generation on the grid the pricing drops encouraging prosumers to generate less and consumer to consume more. The payout function g(.) , utilizes a normalization component in the denominator to account for over or under generation distributing the payout along the curve. The payment is at its highest when generation meets the total demand and at its lowest as generation starts to saturate the market because of low demand.

$$g(x, t_p, t_c) = \frac{x^n * q_{t_p=t_c}}{e^{\frac{(t_p-t_c)^2}{a}}}$$

In [116…

```
import math
# NRGXChange Payment g(.) Function
def g(price,p,tp,tc,a,n):
    x = p
    q = (0.01*price)
```

```python
    try:
        #print(f"n{n},tp{tp},tc{tc},a{a},q{q}")
        pay = abs((pow(x,n)*q)/math.exp(pow((tp-tc),2)/a))
    except OverflowError:
        pay = float('inf')
    return pay
```

Consumer Charge/Cost Function

Where x, is the net energy of the prosumer. q, is the maximum price allowed. t_p ,is the total produced energy of all prosumers. t_c is the total consumption of all the prosumers. a , is a scaling constant to adjust the pay out. Similarly lets consider the cost of energy for consumers to purchase based on pricing set by the h(.) function. In tandem these incentives are non-linear because of the distribution curve. The shape of that curve can be adjusted to the size of the network and the volatility of the network.

$$h(y, t_p, t_c) = \frac{y * r_{t_c >> t_p} * t_c}{t_c + t_p}$$

Where y is the withdrawn energy, and $r_{t_c >> t_p}$ is the maximum cost of energy delivered by the utility when the energy supply by prosumers is low. Again, t_p is the total production and t_c is the total consumption of the prosumers in the network. The minimum payment by the utility in the historical payment prices would indicate the minimum amount willing to charge customers for energy in order to cover the cost of delivering the energy. We will use the minimum price in our list for r.

In [117…

```python
# NRGXChange Charge h(.) Function
def h(price,c,tp,tc):
    y = c
    r = (0.01*price)
    try:
        cost = (y*r*tc)/(tc+tp)
    except OverflowError:
        cost = float('inf')
    return cost
```

Apply Payments and Charges to Prosumers Using NRG-X-Change

The scalar for the nrg payment would need to be self-tracking. A sweep of possible a values was done to track when any of the value goes past the "inf" value.

In [119…

```python
####################################################################
# Calculate and update the dataframe with nrg payments at time t
####################################################################
```

```
def get_nrg_payments(df_by_t,max_price,min_price,a=0,n=1):
    for t in df_by_t:
        tc = t['consumption'].sum()
        tp = abs(t['net_energy']).sum()
        t['nrg_v'] = t['net_energy'].apply(lambda x: g(price=max_price,p=abs(x),tc=tc,tp=tp,n=n,a=a) if abs(x) > 0 else (
        t['prosumer_debit'] = t['consumption'].apply(lambda x: -(h(price=min_price,c=x,tc=tc,tp=tp)) if abs(x) > 0 else (
        t['prosumer_revenue'] = t['nrg_v'] + t['prosumer_debit']
    return df_by_t
```

In [120…

```
def apply_nrg_payments(trials,n=1):
    for trial in trials:
        ####################################################################
        # Apply NRG-X change payments for each time 't'
        ####################################################################
        prosumers_n = trial['prosumers_n']
        prosumers_n_t =trial['prosumers_n_t']
        # Set historical pricing limits for NRG
        max_price = prosumers_n['price'].max()
        min_price = prosumers_n['price'].min()
        # Applying payments to each record
        prev_std = 0
        a_scaled = 0
        # Scaling the 'a' until payments are no longer sensitive
        for a in np.arange(start=10000,stop=(10000*1000),step=10000):
            prosumers_n_t = get_nrg_payments(prosumers_n_t,max_price,min_price,a=a,n=n)
            df = pd.concat(prosumers_n_t)
            std = df['nrg_v'].std()
            pct_c = ((std - prev_std)/std)
            prev_std = std
            if  pct_c < 0.001:
                a_scaled = a
                break
        # store the scaled a value for this trail given N proumers
        trial['a_scaled'] = a_scaled
    return trials
```

Shapley Value Method

Review of Game Theory and Shapley Value

The game is in terms of a **characteristic function**, which specfies for every group of players the total payoff that the members of S can by signing an greement among themselves; this payoff is available for distribution among the members of the group. A coalitional game with transferable payoff is a pair $< N, v >$ where $N = \{1, \ldots, n\}$ is the set of players and for every subset S of I (called a coalition) $v(S) \in \mathbb{R}$ is the total payoff that is available for division among members of S (called the worth of S). We assume that the larger the coalition the larger the payoff (this property is called superadditivity).

An agreement amongst players is a list $(x_1, x_1, \ldots, x_n)$ where x_1, is the proposed payoff to individual i. Shapley value is interpreted in terms of **expected marginal contribution**. It is calculated by considering all the possible orders of arrival of the players into a room and giving each player his marginal contribution.

In [121…

```
# Shapley Value Python Logic
# Authored by Susobhan Ghosh
# https://github.com/susobhang70
# Committed on 02/01/2020
from itertools import combinations
import bisect
#Create Combinatorial from List
def power_set(List):
    PS = [list(j) for i in range(len(List)) for j in combinations(List, i+1)]
    return PS
#Calculate Shapley from Characteristic Value list
def get_shapley(n,v):
    tempList = list([i for i in range(n)])
    N = power_set(tempList)
    shapley_values = []
    for i in range(n):
        shapley = 0
        for j in N:
            if i not in j:
                cmod = len(j)
                Cui = j[:]
                bisect.insort_left(Cui,i)
                l = N.index(j)
                k = N.index(Cui)
                temp = float(float(v[k]) - float(v[l])) *\
                           float(math.factorial(cmod) * math.factorial(n - cmod - 1)) / float(math.factorial(n))
                shapley += temp
        cmod = 0
        Cui = [i]
        k = N.index(Cui)
        temp = float(v[k]) * float(math.factorial(cmod) * math.factorial(n - cmod - 1)) / float(math.factorial(n))
        shapley += temp
        shapley_values.append(shapley)
    return shapley_values
```

In [122…

```python
####################################################################
# Calcualate the coalitional Pay out at each time step of t
####################################################################
def get_coalitional_payments(df_by_t,max_price,min_price,a=0,n=1):
    for t in df_by_t:
        tc = t['consumption'].sum()
        tp = abs(t['net_energy']).sum()
        # identify number of prosumers at every time step of t,
        # that have provided net_energy > 0
        ids_net_energy_given = t[abs(t['net_energy']) > 0]['id']
        N_c=len(ids_net_energy_given)
        # sum the absolute value of all the net energy given
        # by the indentified IDs
        # abs(t.loc[t['id'].isin(ids_net_energy_given)]['net_energy']).sum()
        t['coalition_v'] = 0
        # if number of members is 1 set it to the characteristic value
        if N_c == 1:
            t['coalition_v'] = t['nrg_v']
        # if number of members is greather than 1 calc shapley
        if N_c > 1:
            List = ids_net_energy_given
            # get a power set with combinatorial elements as a list of lists
            PS = [list(j) for i in range(len(List)) for j in combinations(List, i+1)]
            char_vals = []
            # locate all ids in time step within the powerset, (factorial), and sum up
            for nn in PS:
                contribution = abs(t.loc[t['id'].isin(nn)]['net_energy']).sum()
                char_func_val = g(price=max_price,p=contribution,tc=tc,tp=tp,n=n,a=a)
                char_vals.append(char_func_val)
            # use the number of members in the coalition and the
            # characteristic values to calc shapley
            shapleys = get_shapley(N_c,char_vals)
            # add the individual shappley value to each of the id's that generated energy
            for i in range(N_c):
                t.loc[t.index[ids_net_energy_given.values[i]-1], 'coalition_v'] = shapleys[i]
    return df_by_t
```

In [123…

```python
def apply_coalitional_payments(trials,n=1):
    for trial in trials:
        ####################################################################
        # Apply Coalitional payments for each time 't'
        ####################################################################
        prosumers_n = trial['prosumers_n']
        prosumers_n_t =trial['prosumers_n_t']
        a =trial['a_scaled']
        # Set historical pricing limits for NRG
        max_price = prosumers_n['price'].max()
```

```python
        min_price = prosumers_n['price'].min()
        prosumers_n_t = get_coalitional_payments(prosumers_n_t,max_price,min_price,a=a,n=n)
    return trials
```

In [124…

```python
####################################################################
# NRG & Coalitional payment processing for each trial(N) prosumers
####################################################################
# Y=X^(1) , linear
trials = apply_nrg_payments(trials=trials)
trials = apply_coalitional_payments(trials=trials)
# Visualize/Sample of Data
for trial in trials:
    print(f"\nN={trial['N']}")
    print(trial['prosumers_n_t'][0])
```

```
N=2
         id        time       demand  generation  consumption  net_energy  \
25919     1 2019-10-01   757.174284  698.980517    58.193767         0.0
1235759   2 2019-10-01  1230.621807  503.603134   727.018673         0.0

         price  demand_std  generation_std  generation_mean  demand_mean  \
25919    11.66       220.0           260.0             1300         1100
1235759  11.66       220.0           260.0             1300         1100

         nrg_v  prosumer_debit  prosumer_revenue  coalition_v
25919        0       -5.726267         -5.726267            0
1235759      0      -71.538637        -71.538637            0

N=3
         id        time       demand  generation  consumption  net_energy  \
25919     1 2019-10-01   757.174284  698.980517    58.193767         0.0
1235759   2 2019-10-01  1230.621807  503.603134   727.018673         0.0
2445599   3 2019-10-01  1241.295996  681.009070   560.286926         0.0

         price  demand_std  generation_std  generation_mean  demand_mean  \
25919    11.66       220.0           260.0             1300         1100
1235759  11.66       220.0           260.0             1300         1100
2445599  11.66       220.0           260.0             1300         1100

         nrg_v  prosumer_debit  prosumer_revenue  coalition_v
25919        0       -5.726267         -5.726267            0
1235759      0      -71.538637        -71.538637            0
2445599      0      -55.132233        -55.132233            0

N=4
         id        time       demand  generation  consumption  net_energy  \
25919     1 2019-10-01   757.174284  698.980517    58.193767         0.0
1235759   2 2019-10-01  1230.621807  503.603134   727.018673         0.0
```

```
2445599   3 2019-10-01  1241.295996  681.009070   560.286926          0.0
3655439   4 2019-10-01   843.911097  717.432484   126.478613          0.0

         price  demand_std  generation_std  generation_mean  demand_mean  \
25919    11.66       220.0           260.0             1300         1100
1235759  11.66       220.0           260.0             1300         1100
2445599  11.66       220.0           260.0             1300         1100
3655439  11.66       220.0           260.0             1300         1100

         nrg_v  prosumer_debit  prosumer_revenue  coalition_v
25919        0       -5.726267         -5.726267            0
1235759      0      -71.538637        -71.538637            0
2445599      0      -55.132233        -55.132233            0
3655439      0      -12.445495        -12.445495            0

N=5
         id       time       demand  generation  consumption  net_energy  \
25919     1 2019-10-01   757.174284  698.980517    58.193767         0.0
1235759   2 2019-10-01  1230.621807  503.603134   727.018673         0.0
2445599   3 2019-10-01  1241.295996  681.009070   560.286926         0.0
3655439   4 2019-10-01   843.911097  717.432484   126.478613         0.0
4865279   5 2019-10-01   672.649709  647.969906    24.679803         0.0

         price  demand_std  generation_std  generation_mean  demand_mean  \
25919    11.66       220.0           260.0             1300         1100
1235759  11.66       220.0           260.0             1300         1100
2445599  11.66       220.0           260.0             1300         1100
3655439  11.66       220.0           260.0             1300         1100
4865279  11.66       220.0           260.0             1300         1100

         nrg_v  prosumer_debit  prosumer_revenue  coalition_v
25919        0       -5.726267         -5.726267            0
1235759      0      -71.538637        -71.538637            0
2445599      0      -55.132233        -55.132233            0
3655439      0      -12.445495        -12.445495            0
4865279      0       -2.428493         -2.428493            0

N=6
         id       time       demand  generation  consumption  net_energy  \
25919     1 2019-10-01   757.174284  698.980517    58.193767         0.0
1235759   2 2019-10-01  1230.621807  503.603134   727.018673         0.0
2445599   3 2019-10-01  1241.295996  681.009070   560.286926         0.0
3655439   4 2019-10-01   843.911097  717.432484   126.478613         0.0
4865279   5 2019-10-01   672.649709  647.969906    24.679803         0.0
6075119   6 2019-10-01   693.494570  588.374290   105.120280         0.0

         price  demand_std  generation_std  generation_mean  demand_mean  \
25919    11.66       220.0           260.0             1300         1100
1235759  11.66       220.0           260.0             1300         1100
2445599  11.66       220.0           260.0             1300         1100
3655439  11.66       220.0           260.0             1300         1100
```

```
4865279  11.66       220.0           260.0             1300         1100
6075119  11.66       220.0           260.0             1300         1100

         nrg_v  prosumer_debit  prosumer_revenue  coalition_v
25919        0       -5.726267         -5.726267            0
1235759      0      -71.538637        -71.538637            0
2445599      0      -55.132233        -55.132233            0
3655439      0      -12.445495        -12.445495            0
4865279      0       -2.428493         -2.428493            0
6075119      0      -10.343836        -10.343836            0

N=7
         id       time       demand  generation  consumption  net_energy  \
25919     1 2019-10-01   757.174284  698.980517    58.193767         0.0
1235759   2 2019-10-01  1230.621807  503.603134   727.018673         0.0
2445599   3 2019-10-01  1241.295996  681.009070   560.286926         0.0
3655439   4 2019-10-01   843.911097  717.432484   126.478613         0.0
4865279   5 2019-10-01   672.649709  647.969906    24.679803         0.0
6075119   6 2019-10-01   693.494570  588.374290   105.120280         0.0
7284959   7 2019-10-01   974.670409  725.090088   249.580321         0.0

         price  demand_std  generation_std  generation_mean  demand_mean  \
25919    11.66       220.0           260.0             1300         1100
1235759  11.66       220.0           260.0             1300         1100
2445599  11.66       220.0           260.0             1300         1100
3655439  11.66       220.0           260.0             1300         1100
4865279  11.66       220.0           260.0             1300         1100
6075119  11.66       220.0           260.0             1300         1100
7284959  11.66       220.0           260.0             1300         1100

         nrg_v  prosumer_debit  prosumer_revenue  coalition_v
25919        0       -5.726267         -5.726267            0
1235759      0      -71.538637        -71.538637            0
2445599      0      -55.132233        -55.132233            0
3655439      0      -12.445495        -12.445495            0
4865279      0       -2.428493         -2.428493            0
6075119      0      -10.343836        -10.343836            0
7284959      0      -24.558704        -24.558704            0

N=8
         id       time       demand  generation  consumption  net_energy  \
25919     1 2019-10-01   757.174284  698.980517    58.193767         0.0
1235759   2 2019-10-01  1230.621807  503.603134   727.018673         0.0
2445599   3 2019-10-01  1241.295996  681.009070   560.286926         0.0
3655439   4 2019-10-01   843.911097  717.432484   126.478613         0.0
4865279   5 2019-10-01   672.649709  647.969906    24.679803         0.0
6075119   6 2019-10-01   693.494570  588.374290   105.120280         0.0
7284959   7 2019-10-01   974.670409  725.090088   249.580321         0.0
8494799   8 2019-10-01  1040.271045  894.004707   146.266338         0.0

         price  demand_std  generation_std  generation_mean  demand_mean  \
```

25919	11.66	220.0	260.0	1300	1100
1235759	11.66	220.0	260.0	1300	1100
2445599	11.66	220.0	260.0	1300	1100
3655439	11.66	220.0	260.0	1300	1100
4865279	11.66	220.0	260.0	1300	1100
6075119	11.66	220.0	260.0	1300	1100
7284959	11.66	220.0	260.0	1300	1100
8494799	11.66	220.0	260.0	1300	1100

	nrg_v	prosumer_debit	prosumer_revenue	coalition_v
25919	0	-5.726267	-5.726267	0
1235759	0	-71.538637	-71.538637	0
2445599	0	-55.132233	-55.132233	0
3655439	0	-12.445495	-12.445495	0
4865279	0	-2.428493	-2.428493	0
6075119	0	-10.343836	-10.343836	0
7284959	0	-24.558704	-24.558704	0
8494799	0	-14.392608	-14.392608	0

N=9

	id	time	demand	generation	consumption	net_energy \
25919	1	2019-10-01	757.174284	698.980517	58.193767	0.0
1235759	2	2019-10-01	1230.621807	503.603134	727.018673	0.0
2445599	3	2019-10-01	1241.295996	681.009070	560.286926	0.0
3655439	4	2019-10-01	843.911097	717.432484	126.478613	0.0
4865279	5	2019-10-01	672.649709	647.969906	24.679803	0.0
6075119	6	2019-10-01	693.494570	588.374290	105.120280	0.0
7284959	7	2019-10-01	974.670409	725.090088	249.580321	0.0
8494799	8	2019-10-01	1040.271045	894.004707	146.266338	0.0
9704639	9	2019-10-01	948.540002	695.130495	253.409507	0.0

	price	demand_std	generation_std	generation_mean	demand_mean \
25919	11.66	220.0	260.0	1300	1100
1235759	11.66	220.0	260.0	1300	1100
2445599	11.66	220.0	260.0	1300	1100
3655439	11.66	220.0	260.0	1300	1100
4865279	11.66	220.0	260.0	1300	1100
6075119	11.66	220.0	260.0	1300	1100
7284959	11.66	220.0	260.0	1300	1100
8494799	11.66	220.0	260.0	1300	1100
9704639	11.66	220.0	260.0	1300	1100

	nrg_v	prosumer_debit	prosumer_revenue	coalition_v
25919	0	-5.726267	-5.726267	0
1235759	0	-71.538637	-71.538637	0
2445599	0	-55.132233	-55.132233	0
3655439	0	-12.445495	-12.445495	0
4865279	0	-2.428493	-2.428493	0
6075119	0	-10.343836	-10.343836	0
7284959	0	-24.558704	-24.558704	0
8494799	0	-14.392608	-14.392608	0

```
9704639      0      -24.935496        -24.935496           0

N=10
          id       time        demand  generation  consumption  net_energy  \
25919      1 2019-10-01    757.174284  698.980517    58.193767         0.0
1235759    2 2019-10-01   1230.621807  503.603134   727.018673         0.0
2445599    3 2019-10-01   1241.295996  681.009070   560.286926         0.0
3655439    4 2019-10-01    843.911097  717.432484   126.478613         0.0
4865279    5 2019-10-01    672.649709  647.969906    24.679803         0.0
6075119    6 2019-10-01    693.494570  588.374290   105.120280         0.0
7284959    7 2019-10-01    974.670409  725.090088   249.580321         0.0
8494799    8 2019-10-01   1040.271045  894.004707   146.266338         0.0
9704639    9 2019-10-01    948.540002  695.130495   253.409507         0.0
10914479  10 2019-10-01    766.927736  385.539889   381.387847         0.0

          price  demand_std  generation_std  generation_mean  demand_mean  \
25919     11.66       220.0           260.0             1300         1100
1235759   11.66       220.0           260.0             1300         1100
2445599   11.66       220.0           260.0             1300         1100
3655439   11.66       220.0           260.0             1300         1100
4865279   11.66       220.0           260.0             1300         1100
6075119   11.66       220.0           260.0             1300         1100
7284959   11.66       220.0           260.0             1300         1100
8494799   11.66       220.0           260.0             1300         1100
9704639   11.66       220.0           260.0             1300         1100
10914479  11.66       220.0           260.0             1300         1100

          nrg_v  prosumer_debit  prosumer_revenue  coalition_v
25919         0       -5.726267         -5.726267            0
1235759       0      -71.538637        -71.538637            0
2445599       0      -55.132233        -55.132233            0
3655439       0      -12.445495        -12.445495            0
4865279       0       -2.428493         -2.428493            0
6075119       0      -10.343836        -10.343836            0
7284959       0      -24.558704        -24.558704            0
8494799       0      -14.392608        -14.392608            0
9704639       0      -24.935496        -24.935496            0
10914479      0      -37.528564        -37.528564            0
```

Example of Shapley value Calculation for N Prosumers

We consider a comunity of solar prosumers $P = 1, 2, 3..N$, who agree to form a coalition and produce energy. The number of possible coalitions are 2^n and the number of ways to build the grand coalition is $N!$.

In [125…

```
######################################################################
# Consider Convex , Linear & Concave characteristic functions
######################################################################
# Adjust the value of 'X^n' used in NRG payment function
```

```
X_n= [-2,-0.5,1,0.5,2]
x_n_trials = []
for n in X_n :
    trials = apply_nrg_payments(trials=trials,n=n)
    trials = apply_coalitional_payments(trials=trials,n=n)
    x_n_trials.append({'X_n':n,'trials':trials})
```

Comparison of Shapley value for Convex, Linear and Concave characteristic functions

Energy produced by individual Prosumers

Shapley value calculation with and without coalition for different characteristic functions

In [126…

```
table = {}
for x_n_trial in x_n_trials :
    x_n = x_n_trial['X_n']
    col_title = f'Y=X^({x_n})'
    trials = x_n_trial['trials']
    rows=[]
    for trial in trials:
        df = pd.concat(trial['prosumers_n_t'])
        avg_wo_co = df['nrg_v'].mean()
        avg_w_co = df['coalition_v'].mean()
        rows.append({"X_n":x_n, "Number of Prosumers":trial['N'],"With Coalition":avg_w_co,"Without Coalition":avg_w_co}
    table[col_title] = rows

lines = []
for header in table.keys():
    line=""
    for i in range(len(table[header])):
        for row in table[header]:
            print(row)


print(lines)
```

```
.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 1, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 1, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}
{'X_n': 1, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 1, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 1, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}
{'X_n': 1, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 1, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 1, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}
```

{'X_n': 1, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 1, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 0.5, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}
{'X_n': 0.5, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 0.5, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 0.5, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}
{'X_n': 0.5, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 0.5, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 0.5, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}
{'X_n': 0.5, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 0.5, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 0.5, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}
{'X_n': 0.5, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 0.5, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 0.5, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}
{'X_n': 0.5, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 0.5, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 0.5, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}
{'X_n': 0.5, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 0.5, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 0.5, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}
{'X_n': 0.5, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 0.5, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 0.5, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}
{'X_n': 0.5, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 0.5, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 0.5, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}
{'X_n': 0.5, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 0.5, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 0.5, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}
{'X_n': 0.5, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 0.5, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 0.5, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}
{'X_n': 0.5, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 0.5, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 0.5, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}
{'X_n': 0.5, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 0.5, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 0.5, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}
{'X_n': 0.5, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 0.5, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 0.5, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}
{'X_n': 0.5, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 0.5, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 0.5, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}
{'X_n': 0.5, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 0.5, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 0.5, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}
{'X_n': 0.5, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 0.5, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 0.5, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}

{'X_n': 0.5, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 0.5, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 0.5, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}
{'X_n': 0.5, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 0.5, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 0.5, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}
{'X_n': 0.5, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 0.5, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 0.5, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}
{'X_n': 0.5, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 0.5, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 0.5, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}
{'X_n': 0.5, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 0.5, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 0.5, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}
{'X_n': 0.5, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 0.5, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 0.5, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}
{'X_n': 0.5, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 0.5, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 0.5, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}
{'X_n': 0.5, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 0.5, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 0.5, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}
{'X_n': 0.5, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 0.5, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 0.5, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}
{'X_n': 0.5, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 0.5, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 0.5, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}
{'X_n': 0.5, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 0.5, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 2, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}
{'X_n': 2, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 2, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 2, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}
{'X_n': 2, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 2, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 2, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}
{'X_n': 2, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 2, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 2, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}
{'X_n': 2, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 2, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 2, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}
{'X_n': 2, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 2, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 2, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}
{'X_n': 2, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 2, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 2, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}

{'X_n': 2, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 2, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 2, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}
{'X_n': 2, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 2, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 2, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}
{'X_n': 2, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 2, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 2, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}
{'X_n': 2, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 2, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 2, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}
{'X_n': 2, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 2, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 2, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}
{'X_n': 2, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 2, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 2, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}
{'X_n': 2, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 2, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 2, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}
{'X_n': 2, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 2, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 2, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}
{'X_n': 2, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 2, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 2, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}
{'X_n': 2, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 2, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 2, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}
{'X_n': 2, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 2, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 2, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}
{'X_n': 2, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 2, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 2, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}
{'X_n': 2, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 2, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 2, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}
{'X_n': 2, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 2, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 2, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}
{'X_n': 2, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 2, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 2, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}
{'X_n': 2, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 2, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 2, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}
{'X_n': 2, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 2, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 2, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}

```
{'X_n': 2, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 2, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
{'X_n': 2, 'Number of Prosumers': 2, 'With Coalition': 16668.612473211433, 'Without Coalition': 16668.612473211433}
{'X_n': 2, 'Number of Prosumers': 3, 'With Coalition': 22587.925924124862, 'Without Coalition': 22587.925924124862}
{'X_n': 2, 'Number of Prosumers': 4, 'With Coalition': 23514.918469542215, 'Without Coalition': 23514.918469542215}
{'X_n': 2, 'Number of Prosumers': 5, 'With Coalition': 24173.35492380917, 'Without Coalition': 24173.35492380917}
{'X_n': 2, 'Number of Prosumers': 6, 'With Coalition': 21073.363980216473, 'Without Coalition': 21073.363980216473}
{'X_n': 2, 'Number of Prosumers': 7, 'With Coalition': 15214.08840843334, 'Without Coalition': 15214.08840843334}
{'X_n': 2, 'Number of Prosumers': 8, 'With Coalition': 13931.69348528232, 'Without Coalition': 13931.69348528232}
{'X_n': 2, 'Number of Prosumers': 9, 'With Coalition': 9774.919591173862, 'Without Coalition': 9774.919591173862}
{'X_n': 2, 'Number of Prosumers': 10, 'With Coalition': 5716.662792516543, 'Without Coalition': 5716.662792516543}
[]
```

In [ ]:

```
```

In [ ]:

```
```

/