Create Prosumer Coalitions (N=3,5,10)

Pull in the same prosumers and generate the NRG exchange payment for the given consumption and production. The Total Consumption vs. the Total Production is plotted for each itteration to show the incremental change of the growing prosumer network. The data for each prosumer has been synthesized with the same starting values. The max and min payments prices came from the historial max and min payments from the prosumer service territory. These max and min prices where used in the NRG-X-change mechanisim to calculate the payment for the prosumers and for the consumers. The prosumers that are generating more than they consume are paid according to how much they are providing for the demand of the others on the network at that time step. The time steps are divided by months. The coalitional game is created to join the generation of more than one prosumer before getting paid by the NRG-X-Change mechanisim to improve the odds of payment.

Prosumers N=3

In [20]:
```
# Data gathering and synthesization has been done in a seperate module
import p0_data_gather as p0

# Pull data from a dataset of randomly insantiated prosumer, synthesized from EIA.gov data trends
dem_mean = 1100
gen_mean = 1300
prosumers_n = p0.get_data(path='data/prosumer_N10_all_20210305_1129.csv',query=f'id > 0 & id<=3 & demand_std ==

# Gather Data into Monthly Time Steps
prosumers_n_t = [pd.DataFrame(y) for x, y in prosumers_n.groupby('time', as_index=False)]

# Print Sample of Data
from IPython.display import HTML
HTML(prosumers_n_t[0].to_html(index=False))
```

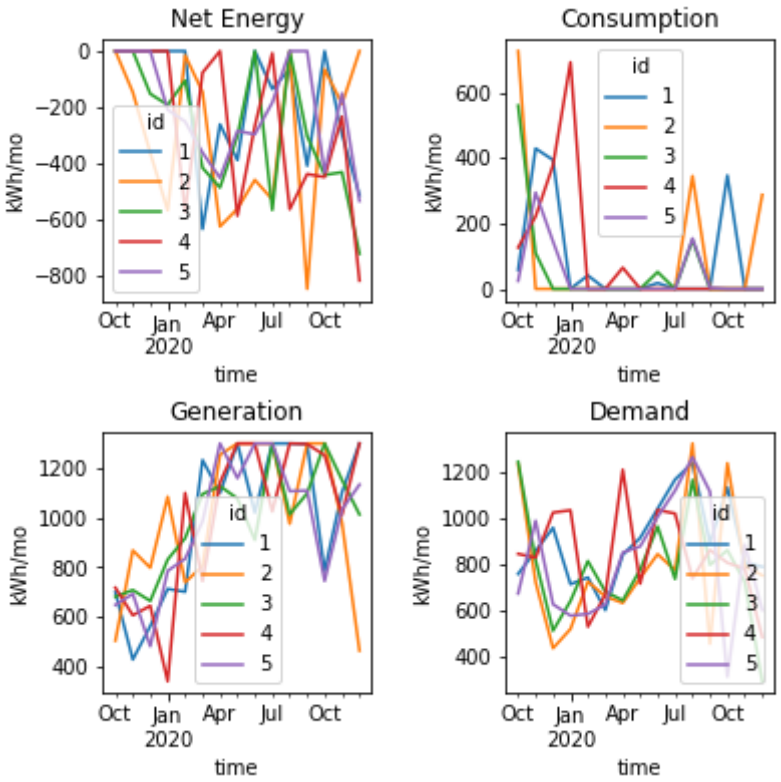
Out[20]:

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean
1	2019-10-01	757.174284	698.980517	58.193767	0.0	11.66	220.0	260.0	1300	1100
2	2019-10-01	1230.621807	503.603134	727.018673	0.0	11.66	220.0	260.0	1300	1100
3	2019-10-01	1241.295996	681.009070	560.286926	0.0	11.66	220.0	260.0	1300	1100

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean
4	2019-10-01	843.911097	717.432484	126.478613	0.0	11.66	220.0	260.0	1300	1100
5	2019-10-01	672.649709	647.969906	24.679803	0.0	11.66	220.0	260.0	1300	1100

In [21]:

```
import numpy as np
import matplotlib.pyplot as plt
df_plt= prosumers_n
fig, axes = plt.subplots(nrows=2, ncols=2,figsize=(6, 6))
plt.subplots_adjust(wspace=0.5, hspace=0.5)
df_plt.pivot(index='time', columns='id', values='net_energy').plot(ax=axes[0, 0])
axes[0, 0].set_title("Net Energy")
axes[0, 0].set_ylabel("kWh/mo")
df_plt.pivot(index='time', columns='id', values='consumption').plot(ax=axes[0, 1])
axes[0, 1].set_title("Consumption")
axes[0, 1].set_ylabel("kWh/mo")
df_plt.pivot(index='time', columns='id', values='generation').plot(ax=axes[1, 0])
axes[1, 0].set_title("Generation")
axes[1, 0].set_ylabel("kWh/mo")
df_plt.pivot(index='time', columns='id', values='demand').plot(ax=axes[1, 1])
axes[1, 1].set_title("Demand")
axes[1, 1].set_ylabel("kWh/mo")
plt.show()
```



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 [22]:

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

```
    x = p
    q = (0.01*price)
    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 [23]:

```
# 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 [24]:
```
import pandas as pd
#calculate the payment at time t, for all prosumers
def get_nrg_payments(df_by_t,max_price,min_price,a=0):
    if a==0:
        a = 100000 #default the a scaler
    for t in df_by_t:
        tc = t['consumption'].sum()
        tp = abs(t['net_energy']).sum()
        t['prosumer_credit'] = t['net_energy'].apply(lambda x: g(price=max_price,p=abs(x),tc=tc,tp=tp,n=1,a=a)
        t['prosumer_debit'] = t['consumption'].apply(lambda x: -(h(price=min_price,c=x,tc=tc,tp=tp)) if abs(x)
        #print(f"tc={tc},tp={tp}")
        t['prosumer_revenue'] = t['prosumer_credit'] + t['prosumer_debit']
    return df_by_t

#historical pricing limits
max_price = prosumers_n['price'].max()
min_price = prosumers_n['price'].min()

# Applying payments to each record
prev_std = 0
a_scaled = 0
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)
    df = pd.concat(prosumers_n_t)
    std = df['prosumer_credit'].std()
    pct_c = ((std - prev_std)/std)
    #print(pct_c)
    prev_std = std
    if  pct_c < 0.001:
        a_scaled = a
        print(f"Found significant 'a' scalar to be :{a} because of a {pct_c*100:.4f}% change in standard deviat
        break

# Print Sample of Data
#from IPython.display import HTML
HTML(prosumers_n_t[0].to_html(index=False))
#df = pd.concat(prosumers_n3_tier1_t)
#print(df)
```

```
Found significant 'a' scalar to be :2090000 because of a 0.0999% change in standard deviation of prosumer reven
ue values
```

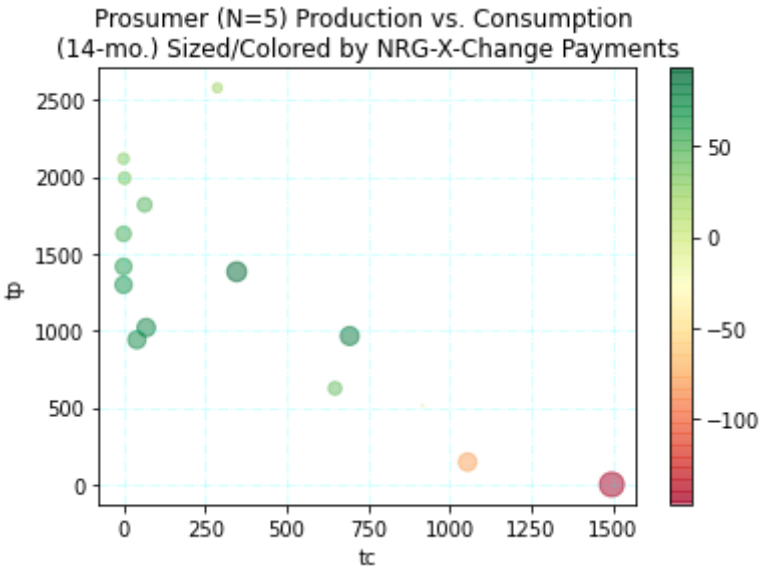
Out[24]:

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean	pr
1	2019-10-01	757.174284	698.980517	58.193767	0.0	11.66	220.0	260.0	1300	1100	

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean	pr
2	2019-10-01	1230.621807	503.603134	727.018673	0.0	11.66	220.0	260.0	1300	1100	
3	2019-10-01	1241.295996	681.009070	560.286926	0.0	11.66	220.0	260.0	1300	1100	
4	2019-10-01	843.911097	717.432484	126.478613	0.0	11.66	220.0	260.0	1300	1100	
5	2019-10-01	672.649709	647.969906	24.679803	0.0	11.66	220.0	260.0	1300	1100	

Analyzing 3 Prosumers An analysis of the collective positive and negative cashflow of the prosumers is shown in the following plot as color. The size of each point indicates the amount of cash and the axis represent the relationship between the total consumption of the group and the total production of the group. Each sample indicates a single point in time, where in this study it ranges for 14 consecutive months.

In [25]:
```
import matplotlib.pyplot as plt
N = len(prosumers_n_t[0])
df = pd.concat(prosumers_n_t)
df = df.groupby(['time']).sum().drop(columns=['id','price'])
df['tp'] = abs(df['net_energy'])
df['tc'] = df['consumption']
fig, ax = plt.subplots()
df.plot(kind='scatter', y='tp', x='tc',s=abs(df['prosumer_revenue']),ax=ax ,alpha=0.5, c= df['prosumer_revenue'
plt.grid(b=True, color='aqua', alpha=0.2, linestyle='dashdot')
plt.title(f"Prosumer (N={N}) Production vs. Consumption \n(14-mo.) Sized/Colored by NRG-X-Change Payments")
plt.show()
```



3. Apply Cooperative Game Theory to Prosumers in a NRG-X-Change Market

3.1 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 [26]:
```
# 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(
        shapley += temp
        shapley_values.append(shapley)
    return shapley_values
```

In [30]:

```
for t in prosumers_n_t:
    tc = t['consumption'].sum()
    tp = abs(t['net_energy']).sum()
    n=1
    a=a_scaled
    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

#Sample of Data
from IPython.display import HTML
HTML(prosumers_n_t[4].to_html(index=False))
```

Out[30]:

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean	p

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean	p
1	2020-02-01	741.830771	700.851761	40.97901	0.000000	11.76	220.0	260.0	1300	1100	
2	2020-02-01	723.828069	738.891573	0.00000	-15.063505	11.76	220.0	260.0	1300	1100	
3	2020-02-01	812.585379	917.494903	0.00000	-104.909524	11.76	220.0	260.0	1300	1100	
4	2020-02-01	529.188718	1100.123378	0.00000	-570.934660	11.76	220.0	260.0	1300	1100	
5	2020-02-01	584.356003	834.353270	0.00000	-249.997266	11.76	220.0	260.0	1300	1100	

In [38]:

```
#calculate coalitional payout each time step of t
for t in prosumers_n_t:
    tc = t['consumption'].sum()
    tp = abs(t['net_energy']).sum()
    a=a_scaled
    #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 N_c == 1:
        t['coalition_v'] = t['nrg_v']
    if N_c > 1:
        List = ids_net_energy_given
        PS = [list(j) for i in range(len(List)) for j in combinations(List, i+1)]
        char_vals = []
        for n in PS:
            #locate all ids in time step within the powerset, (factorial), and sum up
            contribution = abs(t.loc[t['id'].isin(n)]['net_energy']).sum()
            char_func_val = g(price=max_price,p=contribution,tc=tc,tp=tp,n=1,a=a)
            char_vals.append(char_func_val)
        #print(PS)
        #print(char_vals)
        shapleys = get_shapley(N_c,char_vals)
        for i in range(N_c):
            t.loc[t.index[ids_net_energy_given.values[i]-1], 'coalition_v'] = shapleys[i]

#Sample of Data
```

```
from IPython.display import HTML
HTML(prosumers_n_t[3].to_html(index=False))

result = pd.concat(prosumers_n_t)
HTML(result.to_html(index=False))
```

Out[38]:

coalition_v	consumption	demand	demand_mean	demand_std	generation	generation_mean	generation_std	id	net_energy
0.000000	58.193767	757.174284	1100	220.0	698.980517	1300	260.0	1	0.000000
0.000000	727.018673	1230.621807	1100	220.0	503.603134	1300	260.0	2	0.000000
0.000000	560.286926	1241.295996	1100	220.0	681.009070	1300	260.0	3	0.000000
0.000000	126.478613	843.911097	1100	220.0	717.432484	1300	260.0	4	0.000000
0.000000	24.679803	672.649709	1100	220.0	647.969906	1300	260.0	5	0.000000
0.000000	429.241387	856.581711	1100	220.0	427.340324	1300	260.0	1	0.000000
11.883133	0.000000	722.211216	1100	220.0	868.127766	1300	260.0	2	-145.916550
0.000000	109.427498	817.316715	1100	220.0	707.889217	1300	260.0	3	0.000000
0.000000	221.458714	827.408281	1100	220.0	605.949567	1300	260.0	4	0.000000
0.000000	294.528610	986.823771	1100	220.0	692.295160	1300	260.0	5	0.000000
0.000000	394.162348	956.424678	1100	220.0	562.262330	1300	260.0	1	0.000000
40.450701	0.000000	436.158803	1100	220.0	797.827130	1300	260.0	2	-361.668327
16.988558	0.000000	512.060530	1100	220.0	663.954644	1300	260.0	3	-151.894114
0.000000	378.017690	1022.271106	1100	220.0	644.253416	1300	260.0	4	0.000000

coalition_v	consumption	demand	demand_mean	demand_std	generation	generation_mean	generation_std	id	net_energy	
0.000000	144.756695	625.994411	1100	220.0	481.237716	1300	260.0	5	0.000000	
0.000000	0.401549	712.969492	1100	220.0	712.567943	1300	260.0	1	0.000000	
66.007228	0.000000	518.965013	1100	220.0	1084.608756	1300	260.0	2	-565.643743	6
22.315937	0.000000	640.004559	1100	220.0	831.239223	1300	260.0	3	-191.234664	2
0.000000	692.771262	1032.366027	1100	220.0	339.594765	1300	260.0	4	0.000000	
24.311022	0.000000	577.141083	1100	220.0	785.472474	1300	260.0	5	-208.331391	
0.000000	40.979010	741.830771	1100	220.0	700.851761	1300	260.0	1	0.000000	
1.236132	0.000000	723.828069	1100	220.0	738.891573	1300	260.0	2	-15.063505	
8.609022	0.000000	812.585379	1100	220.0	917.494903	1300	260.0	3	-104.909524	
46.851696	0.000000	529.188718	1100	220.0	1100.123378	1300	260.0	4	-570.934660	4
20.515125	0.000000	584.356003	1100	220.0	834.353270	1300	260.0	5	-249.997266	
21.445727	0.000000	600.908845	1100	220.0	1232.947114	1300	260.0	1	-632.038269	2
4.928894	0.000000	660.007831	1100	220.0	805.269837	1300	260.0	2	-145.262005	
14.077518	0.000000	680.838258	1100	220.0	1095.724158	1300	260.0	3	-414.885900	
2.594775	0.000000	667.738498	1100	220.0	744.210478	1300	260.0	4	-76.471981	
12.247418	0.000000	624.732258	1100	220.0	985.682326	1300	260.0	5	-360.950068	
7.224485	0.000000	840.259808	1100	220.0	1100.604230	1300	260.0	1	-260.344422	

coalition_v	consumption	demand	demand_mean	demand_std	generation	generation_mean	generation_std	id	net_energy	
17.286840	0.000000	631.583253	1100	220.0	1254.538710	1300	260.0	2	-622.955457	1
13.456672	0.000000	642.718380	1100	220.0	1127.648368	1300	260.0	3	-484.929988	1
0.000000	64.688959	1207.234188	1100	220.0	1142.545229	1300	260.0	4	0.000000	
12.495497	0.000000	849.707278	1100	220.0	1300.000000	1300	260.0	5	-450.292722	1
5.474676	0.000000	912.760967	1100	220.0	1300.000000	1300	260.0	1	-387.239033	
7.921189	0.000000	739.712334	1100	220.0	1300.000000	1300	260.0	2	-560.287666	
4.268162	0.000000	775.732607	1100	220.0	1077.631521	1300	260.0	3	-301.898914	
8.269025	0.000000	715.108925	1100	220.0	1300.000000	1300	260.0	4	-584.891075	
4.008837	0.000000	876.251881	1100	220.0	1159.808066	1300	260.0	5	-283.556185	
0.000000	17.802440	1038.681500	1100	220.0	1020.879060	1300	260.0	1	0.000000	
35.904843	0.000000	842.573368	1100	220.0	1300.000000	1300	260.0	2	-457.426632	3
0.000000	51.700212	962.018220	1100	220.0	910.318008	1300	260.0	3	0.000000	
20.918044	0.000000	1033.504748	1100	220.0	1300.000000	1300	260.0	4	-266.495252	2
23.212677	0.000000	1004.271183	1100	220.0	1300.000000	1300	260.0	5	-295.728817	2
6.242867	0.000000	1165.090456	1100	220.0	1300.000000	1300	260.0	1	-134.909544	
24.398097	0.000000	772.752521	1100	220.0	1300.000000	1300	260.0	2	-527.247479	2
26.112849	0.000000	735.696415	1100	220.0	1300.000000	1300	260.0	3	-564.303585	2

coalition_v	consumption	demand	demand_mean	demand_std	generation	generation_mean	generation_std	id	net_energy	
0.345430	0.000000	1017.990935	1100	220.0	1025.455744	1300	260.0	4	-7.464809	
8.460107	0.000000	1117.175491	1100	220.0	1300.000000	1300	260.0	5	-182.824509	
7.649617	0.000000	1236.711727	1100	220.0	1300.000000	1300	260.0	1	-63.288273	
0.000000	344.981457	1320.799390	1100	220.0	975.817933	1300	260.0	2	0.000000	
0.000000	148.877122	1162.721526	1100	220.0	1013.844404	1300	260.0	3	0.000000	
67.919356	0.000000	738.076595	1100	220.0	1300.000000	1300	260.0	4	-561.923405	6
0.000000	154.346494	1261.774214	1100	220.0	1107.427719	1300	260.0	5	0.000000	
7.430543	0.000000	891.556610	1100	220.0	1298.843324	1300	260.0	1	-407.286714	
15.410766	0.000000	455.297268	1100	220.0	1300.000000	1300	260.0	2	-844.702732	
5.518739	0.000000	796.070701	1100	220.0	1098.566634	1300	260.0	3	-302.495932	
7.969893	0.000000	860.703025	1100	220.0	1297.552831	1300	260.0	4	-436.849806	
0.000000	3.256915	1112.831083	1100	220.0	1109.574168	1300	260.0	5	0.000000	
0.000000	346.839307	1130.832481	1100	220.0	783.993174	1300	260.0	1	0.000000	
4.664741	0.000000	1235.486678	1100	220.0	1300.000000	1300	260.0	2	-64.513322	
31.817195	0.000000	859.968470	1100	220.0	1300.000000	1300	260.0	3	-440.031530	
32.350888	0.000000	805.362242	1100	220.0	1252.774736	1300	260.0	4	-447.412494	3
31.192752	0.000000	313.363713	1100	220.0	744.759198	1300	260.0	5	-431.395485	

coalition_v	consumption	demand	demand_mean	demand_std	generation	generation_mean	generation_std	id	net_energy
15.994812	0.000000	806.965579	1100	220.0	1103.241179	1300	260.0	1	-296.275600
10.161202	0.000000	790.826705	1100	220.0	979.045000	1300	260.0	2	-188.218295
23.226565	0.000000	722.210544	1100	220.0	1152.441576	1300	260.0	3	-430.231032
12.567815	0.000000	779.390564	1100	220.0	1012.187101	1300	260.0	4	-232.796536
8.128461	0.000000	883.079338	1100	220.0	1033.644700	1300	260.0	5	-150.565363
5.016254	0.000000	788.236454	1100	220.0	1300.000000	1300	260.0	1	-511.763546
0.000000	287.520793	750.823892	1100	220.0	463.303099	1300	260.0	2	0.000000
7.054937	0.000000	293.502601	1100	220.0	1013.254735	1300	260.0	3	-719.752134
7.990066	0.000000	484.845007	1100	220.0	1300.000000	1300	260.0	4	-815.154993
5.217866	0.000000	601.377546	1100	220.0	1133.709775	1300	260.0	5	-532.332230

Prosumers N=5

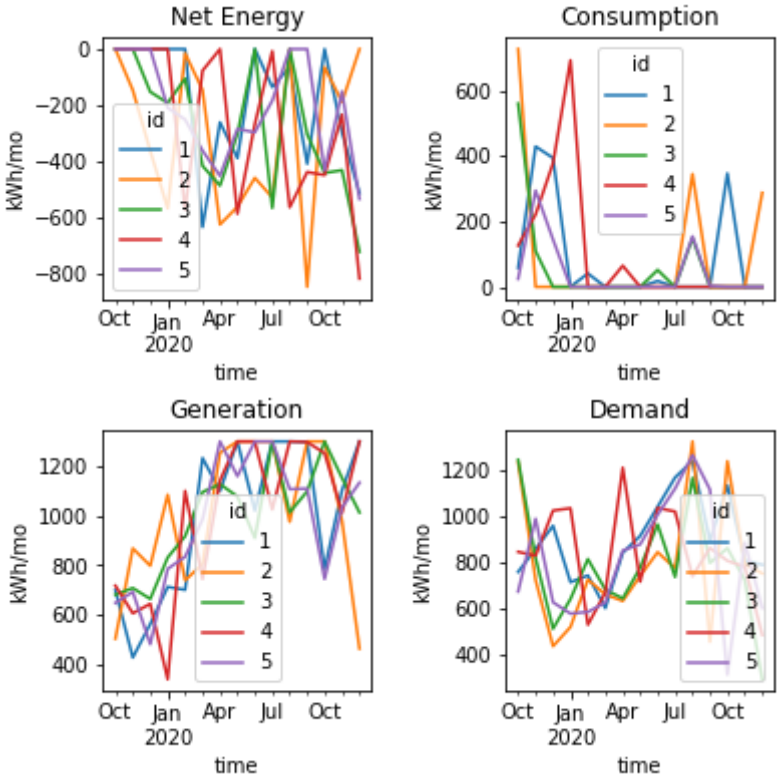
In [39]:

```
# Data gathering and synthesization has been done in a seperate module
import p0_data_gather as p0

# Pull data from a dataset of randomly insantiated prosumer, synthesized from EIA.gov data trends
dem_mean = 1100
gen_mean = 1300
prosumers_n = p0.get_data(path='data/prosumer_N10_all_20210305_1129.csv',query=f'id > 0 & id<=5 & demand_std ==

# Gather Data into Monthly Time Steps
prosumers_n_t = [pd.DataFrame(y) for x, y in prosumers_n.groupby('time', as_index=False)]

# Print Sample of Data
from IPython.display import HTML
```

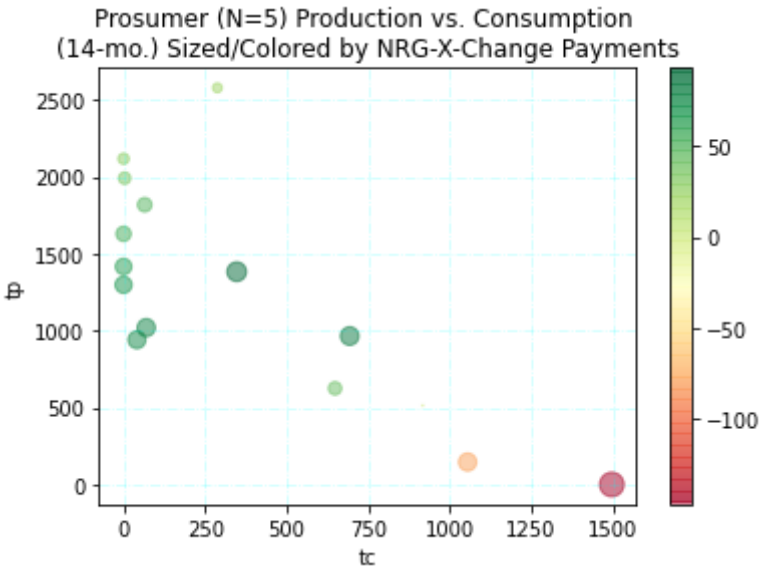
```
HTML(prosumers_n_t[0].to_html(index=False))

import numpy as np
import matplotlib.pyplot as plt
df_plt= prosumers_n
fig, axes = plt.subplots(nrows=2, ncols=2,figsize=(6, 6))
plt.subplots_adjust(wspace=0.5, hspace=0.5)
df_plt.pivot(index='time', columns='id', values='net_energy').plot(ax=axes[0, 0])
axes[0, 0].set_title("Net Energy")
axes[0, 0].set_ylabel("kWh/mo")
df_plt.pivot(index='time', columns='id', values='consumption').plot(ax=axes[0, 1])
axes[0, 1].set_title("Consumption")
axes[0, 1].set_ylabel("kWh/mo")
df_plt.pivot(index='time', columns='id', values='generation').plot(ax=axes[1, 0])
axes[1, 0].set_title("Generation")
axes[1, 0].set_ylabel("kWh/mo")
df_plt.pivot(index='time', columns='id', values='demand').plot(ax=axes[1, 1])
axes[1, 1].set_title("Demand")
axes[1, 1].set_ylabel("kWh/mo")
plt.show()


#historical pricing limits
max_price = prosumers_n['price'].max()
min_price = prosumers_n['price'].min()

# Applying payments to each record
prev_std = 0
a_scaled = 0
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)
    df = pd.concat(prosumers_n_t)
    std = df['prosumer_credit'].std()
    pct_c = ((std - prev_std)/std)
    #print(pct_c)
    prev_std = std
    if  pct_c < 0.001:
        a_scaled = a
        print(f"Found significant 'a' scalar to be :{a} because of a {pct_c*100:.4f}% change in standard deviat
        break

# Print Sample of Data
#from IPython.display import HTML
HTML(prosumers_n_t[0].to_html(index=False))
#df = pd.concat(prosumers_n3_tier1_t)
#print(df)
```

```
import matplotlib.pyplot as plt
N = len(prosumers_n_t[0])
df = pd.concat(prosumers_n_t)
df = df.groupby(['time']).sum().drop(columns=['id','price'])
df['tp'] = abs(df['net_energy'])
df['tc'] = df['consumption']
fig, ax = plt.subplots()
df.plot(kind='scatter', y='tp', x='tc',s=abs(df['prosumer_revenue']),ax=ax ,alpha=0.5, c= df['prosumer_revenue'
plt.grid(b=True, color='aqua', alpha=0.2, linestyle='dashdot')
plt.title(f"Prosumer (N={N}) Production vs. Consumption \n(14-mo.) Sized/Colored by NRG-X-Change Payments")
plt.show()


for t in prosumers_n_t:
    tc = t['consumption'].sum()
    tp = abs(t['net_energy']).sum()
    n=1
    a=a_scaled
    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

#Sample of Data
from IPython.display import HTML
HTML(prosumers_n_t[4].to_html(index=False))

#calculate coalitional payout each time step of t
for t in prosumers_n_t:
    tc = t['consumption'].sum()
    tp = abs(t['net_energy']).sum()
    a=a_scaled
    #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 N_c == 1:
        t['coalition_v'] = t['nrg_v']
    if N_c > 1:
        List = ids_net_energy_given
        PS = [list(j) for i in range(len(List)) for j in combinations(List, i+1)]
        char_vals = []
        for n in PS:
            #locate all ids in time step within the powerset, (factorial), and sum up
            contribution = abs(t.loc[t['id'].isin(n)]['net_energy']).sum()
```

```
            char_func_val = g(price=max_price,p=contribution,tc=tc,tp=tp,n=1,a=a)
            char_vals.append(char_func_val)
        #print(PS)
        #print(char_vals)
        shapleys = get_shapley(N_c,char_vals)
        for i in range(N_c):
            t.loc[t.index[ids_net_energy_given.values[i]-1], 'coalition_v'] = shapleys[i]

#Sample of Data
from IPython.display import HTML
HTML(prosumers_n_t[3].to_html(index=False))

result = pd.concat(prosumers_n_t)
HTML(result.to_html(index=False))
```




```
Found significant 'a' scalar to be :2090000 because of a 0.0999% change in standard deviation of prosumer reven
ue values
```



Out[39]:

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean
1	2019-10-01	757.174284	698.980517	58.193767	0.000000	11.66	220.0	260.0	1300	1100
2	2019-10-01	1230.621807	503.603134	727.018673	0.000000	11.66	220.0	260.0	1300	1100
3	2019-10-01	1241.295996	681.009070	560.286926	0.000000	11.66	220.0	260.0	1300	1100
4	2019-10-01	843.911097	717.432484	126.478613	0.000000	11.66	220.0	260.0	1300	1100
5	2019-10-01	672.649709	647.969906	24.679803	0.000000	11.66	220.0	260.0	1300	1100
1	2019-11-01	856.581711	427.340324	429.241387	0.000000	12.09	220.0	260.0	1300	1100
2	2019-11-01	722.211216	868.127766	0.000000	-145.916550	12.09	220.0	260.0	1300	1100
3	2019-11-01	817.316715	707.889217	109.427498	0.000000	12.09	220.0	260.0	1300	1100
4	2019-11-01	827.408281	605.949567	221.458714	0.000000	12.09	220.0	260.0	1300	1100

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean
5	2019-11-01	986.823771	692.295160	294.528610	0.000000	12.09	220.0	260.0	1300	1100
1	2019-12-01	956.424678	562.262330	394.162348	0.000000	11.62	220.0	260.0	1300	1100
2	2019-12-01	436.158803	797.827130	0.000000	-361.668327	11.62	220.0	260.0	1300	1100
3	2019-12-01	512.060530	663.954644	0.000000	-151.894114	11.62	220.0	260.0	1300	1100
4	2019-12-01	1022.271106	644.253416	378.017690	0.000000	11.62	220.0	260.0	1300	1100
5	2019-12-01	625.994411	481.237716	144.756695	0.000000	11.62	220.0	260.0	1300	1100
1	2020-01-01	712.969492	712.567943	0.401549	0.000000	11.72	220.0	260.0	1300	1100
2	2020-01-01	518.965013	1084.608756	0.000000	-565.643743	11.72	220.0	260.0	1300	1100
3	2020-01-01	640.004559	831.239223	0.000000	-191.234664	11.72	220.0	260.0	1300	1100
4	2020-01-01	1032.366027	339.594765	692.771262	0.000000	11.72	220.0	260.0	1300	1100
5	2020-01-01	577.141083	785.472474	0.000000	-208.331391	11.72	220.0	260.0	1300	1100
1	2020-02-01	741.830771	700.851761	40.979010	0.000000	11.76	220.0	260.0	1300	1100
2	2020-02-01	723.828069	738.891573	0.000000	-15.063505	11.76	220.0	260.0	1300	1100
3	2020-02-01	812.585379	917.494903	0.000000	-104.909524	11.76	220.0	260.0	1300	1100
4	2020-02-01	529.188718	1100.123378	0.000000	-570.934660	11.76	220.0	260.0	1300	1100
5	2020-02-01	584.356003	834.353270	0.000000	-249.997266	11.76	220.0	260.0	1300	1100
1	2020-03-01	600.908845	1232.947114	0.000000	-632.038269	11.63	220.0	260.0	1300	1100

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean
2	2020-03-01	660.007831	805.269837	0.000000	-145.262005	11.63	220.0	260.0	1300	1100
3	2020-03-01	680.838258	1095.724158	0.000000	-414.885900	11.63	220.0	260.0	1300	1100
4	2020-03-01	667.738498	744.210478	0.000000	-76.471981	11.63	220.0	260.0	1300	1100
5	2020-03-01	624.732258	985.682326	0.000000	-360.950068	11.63	220.0	260.0	1300	1100
1	2020-04-01	840.259808	1100.604230	0.000000	-260.344422	11.71	220.0	260.0	1300	1100
2	2020-04-01	631.583253	1254.538710	0.000000	-622.955457	11.71	220.0	260.0	1300	1100
3	2020-04-01	642.718380	1127.648368	0.000000	-484.929988	11.71	220.0	260.0	1300	1100
4	2020-04-01	1207.234188	1142.545229	64.688959	0.000000	11.71	220.0	260.0	1300	1100
5	2020-04-01	849.707278	1300.000000	0.000000	-450.292722	11.71	220.0	260.0	1300	1100
1	2020-05-01	912.760967	1300.000000	0.000000	-387.239033	9.84	220.0	260.0	1300	1100
2	2020-05-01	739.712334	1300.000000	0.000000	-560.287666	9.84	220.0	260.0	1300	1100
3	2020-05-01	775.732607	1077.631521	0.000000	-301.898914	9.84	220.0	260.0	1300	1100
4	2020-05-01	715.108925	1300.000000	0.000000	-584.891075	9.84	220.0	260.0	1300	1100
5	2020-05-01	876.251881	1159.808066	0.000000	-283.556185	9.84	220.0	260.0	1300	1100
1	2020-06-01	1038.681500	1020.879060	17.802440	0.000000	11.53	220.0	260.0	1300	1100
2	2020-06-01	842.573368	1300.000000	0.000000	-457.426632	11.53	220.0	260.0	1300	1100
3	2020-06-01	962.018220	910.318008	51.700212	0.000000	11.53	220.0	260.0	1300	1100

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean
4	2020-06-01	1033.504748	1300.000000	0.000000	-266.495252	11.53	220.0	260.0	1300	1100
5	2020-06-01	1004.271183	1300.000000	0.000000	-295.728817	11.53	220.0	260.0	1300	1100
1	2020-07-01	1165.090456	1300.000000	0.000000	-134.909544	11.71	220.0	260.0	1300	1100
2	2020-07-01	772.752521	1300.000000	0.000000	-527.247479	11.71	220.0	260.0	1300	1100
3	2020-07-01	735.696415	1300.000000	0.000000	-564.303585	11.71	220.0	260.0	1300	1100
4	2020-07-01	1017.990935	1025.455744	0.000000	-7.464809	11.71	220.0	260.0	1300	1100
5	2020-07-01	1117.175491	1300.000000	0.000000	-182.824509	11.71	220.0	260.0	1300	1100
1	2020-08-01	1236.711727	1300.000000	0.000000	-63.288273	11.61	220.0	260.0	1300	1100
2	2020-08-01	1320.799390	975.817933	344.981457	0.000000	11.61	220.0	260.0	1300	1100
3	2020-08-01	1162.721526	1013.844404	148.877122	0.000000	11.61	220.0	260.0	1300	1100
4	2020-08-01	738.076595	1300.000000	0.000000	-561.923405	11.61	220.0	260.0	1300	1100
5	2020-08-01	1261.774214	1107.427719	154.346494	0.000000	11.61	220.0	260.0	1300	1100
1	2020-09-01	891.556610	1298.843324	0.000000	-407.286714	11.97	220.0	260.0	1300	1100
2	2020-09-01	455.297268	1300.000000	0.000000	-844.702732	11.97	220.0	260.0	1300	1100
3	2020-09-01	796.070701	1098.566634	0.000000	-302.495932	11.97	220.0	260.0	1300	1100
4	2020-09-01	860.703025	1297.552831	0.000000	-436.849806	11.97	220.0	260.0	1300	1100
5	2020-09-01	1112.831083	1109.574168	3.256915	0.000000	11.97	220.0	260.0	1300	1100

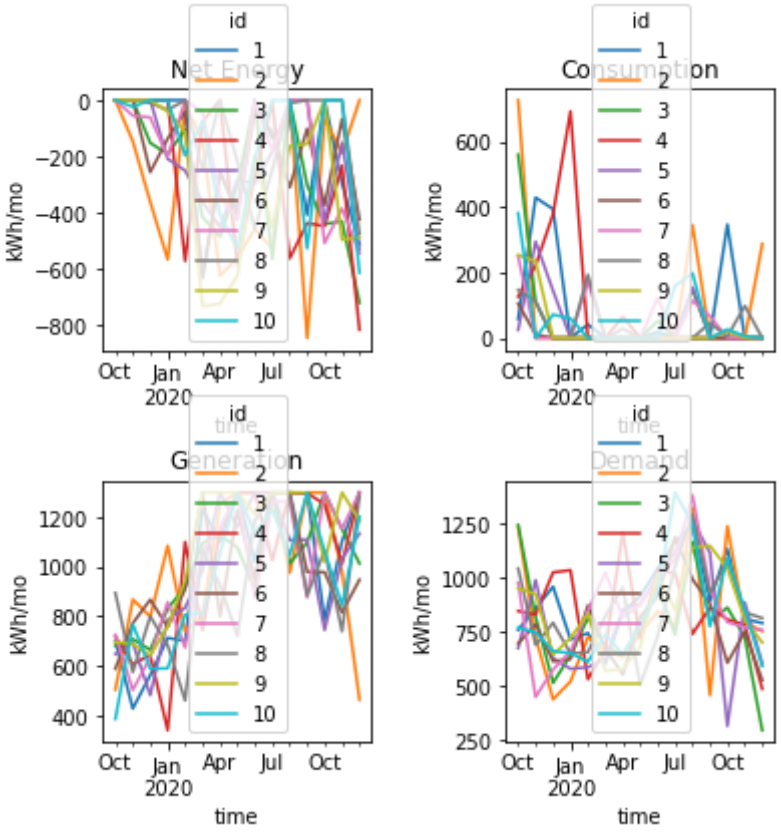
id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean
1	2020-10-01	1130.832481	783.993174	346.839307	0.000000	11.71	220.0	260.0	1300	1100
2	2020-10-01	1235.486678	1300.000000	0.000000	-64.513322	11.71	220.0	260.0	1300	1100
3	2020-10-01	859.968470	1300.000000	0.000000	-440.031530	11.71	220.0	260.0	1300	1100
4	2020-10-01	805.362242	1252.774736	0.000000	-447.412494	11.71	220.0	260.0	1300	1100
5	2020-10-01	313.363713	744.759198	0.000000	-431.395485	11.71	220.0	260.0	1300	1100
1	2020-11-01	806.965579	1103.241179	0.000000	-296.275600	12.00	220.0	260.0	1300	1100
2	2020-11-01	790.826705	979.045000	0.000000	-188.218295	12.00	220.0	260.0	1300	1100
3	2020-11-01	722.210544	1152.441576	0.000000	-430.231032	12.00	220.0	260.0	1300	1100
4	2020-11-01	779.390564	1012.187101	0.000000	-232.796536	12.00	220.0	260.0	1300	1100
5	2020-11-01	883.079338	1033.644700	0.000000	-150.565363	12.00	220.0	260.0	1300	1100
1	2020-12-01	788.236454	1300.000000	0.000000	-511.763546	11.86	220.0	260.0	1300	1100
2	2020-12-01	750.823892	463.303099	287.520793	0.000000	11.86	220.0	260.0	1300	1100
3	2020-12-01	293.502601	1013.254735	0.000000	-719.752134	11.86	220.0	260.0	1300	1100
4	2020-12-01	484.845007	1300.000000	0.000000	-815.154993	11.86	220.0	260.0	1300	1100
5	2020-12-01	601.377546	1133.709775	0.000000	-532.332230	11.86	220.0	260.0	1300	1100

Prosumers N=10

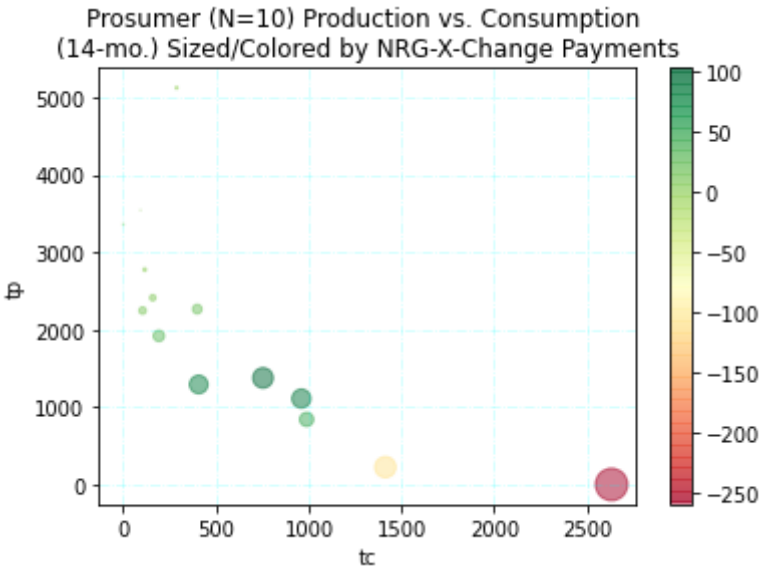
In [40]:
```
# Data gathering and synthesization has been done in a seperate module
import p0_data_gather as p0

# Pull data from a dataset of randomly insantiated prosumer, synthesized from EIA.gov data trends
dem_mean = 1100
gen_mean = 1300
prosumers_n = p0.get_data(path='data/prosumer_N10_all_20210305_1129.csv',query=f'id > 0 & id<=10 & demand_std =

# Gather Data into Monthly Time Steps
prosumers_n_t = [pd.DataFrame(y) for x, y in prosumers_n.groupby('time', as_index=False)]

# Print Sample of Data
from IPython.display import HTML
HTML(prosumers_n_t[0].to_html(index=False))

import numpy as np
import matplotlib.pyplot as plt
df_plt= prosumers_n
fig, axes = plt.subplots(nrows=2, ncols=2,figsize=(6, 6))
plt.subplots_adjust(wspace=0.5, hspace=0.5)
df_plt.pivot(index='time', columns='id', values='net_energy').plot(ax=axes[0, 0])
axes[0, 0].set_title("Net Energy")
axes[0, 0].set_ylabel("kWh/mo")
df_plt.pivot(index='time', columns='id', values='consumption').plot(ax=axes[0, 1])
axes[0, 1].set_title("Consumption")
axes[0, 1].set_ylabel("kWh/mo")
df_plt.pivot(index='time', columns='id', values='generation').plot(ax=axes[1, 0])
axes[1, 0].set_title("Generation")
axes[1, 0].set_ylabel("kWh/mo")
df_plt.pivot(index='time', columns='id', values='demand').plot(ax=axes[1, 1])
axes[1, 1].set_title("Demand")
axes[1, 1].set_ylabel("kWh/mo")
plt.show()


#historical pricing limits
max_price = prosumers_n['price'].max()
min_price = prosumers_n['price'].min()

# Applying payments to each record
prev_std = 0
a_scaled = 0
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)
    df = pd.concat(prosumers_n_t)
    std = df['prosumer_credit'].std()
```

```
    pct_c = ((std - prev_std)/std)
    #print(pct_c)
    prev_std = std
    if  pct_c < 0.001:
        a_scaled = a
        print(f"Found significant 'a' scalar to be :{a} because of a {pct_c*100:.4f}% change in standard deviat
        break

# Print Sample of Data
#from IPython.display import HTML
HTML(prosumers_n_t[0].to_html(index=False))
#df = pd.concat(prosumers_n3_tier1_t)
#print(df)


import matplotlib.pyplot as plt
N = len(prosumers_n_t[0])
df = pd.concat(prosumers_n_t)
df = df.groupby(['time']).sum().drop(columns=['id','price'])
df['tp'] = abs(df['net_energy'])
df['tc'] = df['consumption']
fig, ax = plt.subplots()
df.plot(kind='scatter', y='tp', x='tc',s=abs(df['prosumer_revenue']),ax=ax ,alpha=0.5, c= df['prosumer_revenue'
plt.grid(b=True, color='aqua', alpha=0.2, linestyle='dashdot')
plt.title(f"Prosumer (N={N}) Production vs. Consumption \n(14-mo.) Sized/Colored by NRG-X-Change Payments")
plt.show()


for t in prosumers_n_t:
    tc = t['consumption'].sum()
    tp = abs(t['net_energy']).sum()
    n=1
    a=a_scaled
    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

#Sample of Data
from IPython.display import HTML
HTML(prosumers_n_t[4].to_html(index=False))

#calculate coalitional payout each time step of t
for t in prosumers_n_t:
    tc = t['consumption'].sum()
    tp = abs(t['net_energy']).sum()
    a=a_scaled
    #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 N_c == 1:
        t['coalition_v'] = t['nrg_v']
    if N_c > 1:
        List = ids_net_energy_given
        PS = [list(j) for i in range(len(List)) for j in combinations(List, i+1)]
        char_vals = []
        for n in PS:
            #locate all ids in time step within the powerset, (factorial), and sum up
            contribution = abs(t.loc[t['id'].isin(n)]['net_energy']).sum()
            char_func_val = g(price=max_price,p=contribution,tc=tc,tp=tp,n=1,a=a)
            char_vals.append(char_func_val)
        #print(PS)
        #print(char_vals)
        shapleys = get_shapley(N_c,char_vals)
        for i in range(N_c):
            t.loc[t.index[ids_net_energy_given.values[i]-1], 'coalition_v'] = shapleys[i]

#Sample of Data
from IPython.display import HTML
HTML(prosumers_n_t[3].to_html(index=False))

result = pd.concat(prosumers_n_t)
HTML(result.to_html(index=False))
```




```
Found significant 'a' scalar to be :1510000 because of a 0.0998% change in standard deviation of prosumer reven
ue values
```



Out[40]:

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean
1	2019-10-01	757.174284	698.980517	58.193767	0.000000	11.66	220.0	260.0	1300	1100
2	2019-10-01	1230.621807	503.603134	727.018673	0.000000	11.66	220.0	260.0	1300	1100
3	2019-10-01	1241.295996	681.009070	560.286926	0.000000	11.66	220.0	260.0	1300	1100
4	2019-10-01	843.911097	717.432484	126.478613	0.000000	11.66	220.0	260.0	1300	1100
5	2019-10-01	672.649709	647.969906	24.679803	0.000000	11.66	220.0	260.0	1300	1100
6	2019-10-01	693.494570	588.374290	105.120280	0.000000	11.66	220.0	260.0	1300	1100
7	2019-10-01	974.670409	725.090088	249.580321	0.000000	11.66	220.0	260.0	1300	1100
8	2019-10-01	1040.271045	894.004707	146.266338	0.000000	11.66	220.0	260.0	1300	1100
9	2019-10-01	948.540002	695.130495	253.409507	0.000000	11.66	220.0	260.0	1300	1100

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean
10	2019-10-01	766.927736	385.539889	381.387847	0.000000	11.66	220.0	260.0	1300	1100
1	2019-11-01	856.581711	427.340324	429.241387	0.000000	12.09	220.0	260.0	1300	1100
2	2019-11-01	722.211216	868.127766	0.000000	-145.916550	12.09	220.0	260.0	1300	1100
3	2019-11-01	817.316715	707.889217	109.427498	0.000000	12.09	220.0	260.0	1300	1100
4	2019-11-01	827.408281	605.949567	221.458714	0.000000	12.09	220.0	260.0	1300	1100
5	2019-11-01	986.823771	692.295160	294.528610	0.000000	12.09	220.0	260.0	1300	1100
6	2019-11-01	782.905659	772.206093	10.699567	0.000000	12.09	220.0	260.0	1300	1100
7	2019-11-01	447.009064	502.114731	0.000000	-55.105668	12.09	220.0	260.0	1300	1100
8	2019-11-01	687.682979	574.623116	113.059864	0.000000	12.09	220.0	260.0	1300	1100
9	2019-11-01	922.039250	687.623788	234.415462	0.000000	12.09	220.0	260.0	1300	1100
10	2019-11-01	745.221276	768.429754	0.000000	-23.208479	12.09	220.0	260.0	1300	1100
1	2019-12-01	956.424678	562.262330	394.162348	0.000000	11.62	220.0	260.0	1300	1100
2	2019-12-01	436.158803	797.827130	0.000000	-361.668327	11.62	220.0	260.0	1300	1100
3	2019-12-01	512.060530	663.954644	0.000000	-151.894114	11.62	220.0	260.0	1300	1100
4	2019-12-01	1022.271106	644.253416	378.017690	0.000000	11.62	220.0	260.0	1300	1100
5	2019-12-01	625.994411	481.237716	144.756695	0.000000	11.62	220.0	260.0	1300	1100
6	2019-12-01	611.713281	867.306418	0.000000	-255.593137	11.62	220.0	260.0	1300	1100

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean
7	2019-12-01	573.469592	634.411654	0.000000	-60.942062	11.62	220.0	260.0	1300	1100
8	2019-12-01	789.731824	799.869456	0.000000	-10.137632	11.62	220.0	260.0	1300	1100
9	2019-12-01	642.327667	642.461384	0.000000	-0.133717	11.62	220.0	260.0	1300	1100
10	2019-12-01	659.154092	587.803745	71.350348	0.000000	11.62	220.0	260.0	1300	1100
1	2020-01-01	712.969492	712.567943	0.401549	0.000000	11.72	220.0	260.0	1300	1100
2	2020-01-01	518.965013	1084.608756	0.000000	-565.643743	11.72	220.0	260.0	1300	1100
3	2020-01-01	640.004559	831.239223	0.000000	-191.234664	11.72	220.0	260.0	1300	1100
4	2020-01-01	1032.366027	339.594765	692.771262	0.000000	11.72	220.0	260.0	1300	1100
5	2020-01-01	577.141083	785.472474	0.000000	-208.331391	11.72	220.0	260.0	1300	1100
6	2020-01-01	621.511333	758.046211	0.000000	-136.534878	11.72	220.0	260.0	1300	1100
7	2020-01-01	657.208197	857.101557	0.000000	-199.893360	11.72	220.0	260.0	1300	1100
8	2020-01-01	648.426839	682.949102	0.000000	-34.522264	11.72	220.0	260.0	1300	1100
9	2020-01-01	719.260027	759.011678	0.000000	-39.751651	11.72	220.0	260.0	1300	1100
10	2020-01-01	649.634494	589.870317	59.764178	0.000000	11.72	220.0	260.0	1300	1100
1	2020-02-01	741.830771	700.851761	40.979010	0.000000	11.76	220.0	260.0	1300	1100
2	2020-02-01	723.828069	738.891573	0.000000	-15.063505	11.76	220.0	260.0	1300	1100
3	2020-02-01	812.585379	917.494903	0.000000	-104.909524	11.76	220.0	260.0	1300	1100

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean
4	2020-02-01	529.188718	1100.123378	0.000000	-570.934660	11.76	220.0	260.0	1300	1100
5	2020-02-01	584.356003	834.353270	0.000000	-249.997266	11.76	220.0	260.0	1300	1100
6	2020-02-01	869.572795	909.884944	0.000000	-40.312149	11.76	220.0	260.0	1300	1100
7	2020-02-01	845.912615	673.711935	172.200680	0.000000	11.76	220.0	260.0	1300	1100
8	2020-02-01	653.528006	460.267126	193.260881	0.000000	11.76	220.0	260.0	1300	1100
9	2020-02-01	831.611098	945.912195	0.000000	-114.301097	11.76	220.0	260.0	1300	1100
10	2020-02-01	609.541398	805.684439	0.000000	-196.143041	11.76	220.0	260.0	1300	1100
1	2020-03-01	600.908845	1232.947114	0.000000	-632.038269	11.63	220.0	260.0	1300	1100
2	2020-03-01	660.007831	805.269837	0.000000	-145.262005	11.63	220.0	260.0	1300	1100
3	2020-03-01	680.838258	1095.724158	0.000000	-414.885900	11.63	220.0	260.0	1300	1100
4	2020-03-01	667.738498	744.210478	0.000000	-76.471981	11.63	220.0	260.0	1300	1100
5	2020-03-01	624.732258	985.682326	0.000000	-360.950068	11.63	220.0	260.0	1300	1100
6	2020-03-01	672.766573	1300.000000	0.000000	-627.233427	11.63	220.0	260.0	1300	1100
7	2020-03-01	1020.509349	1063.954379	0.000000	-43.445030	11.63	220.0	260.0	1300	1100
8	2020-03-01	819.585318	1068.434871	0.000000	-248.849553	11.63	220.0	260.0	1300	1100
9	2020-03-01	567.718184	1300.000000	0.000000	-732.281816	11.63	220.0	260.0	1300	1100
10	2020-03-01	736.349267	812.251233	0.000000	-75.901966	11.63	220.0	260.0	1300	1100

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean
1	2020-04-01	840.259808	1100.604230	0.000000	-260.344422	11.71	220.0	260.0	1300	1100
2	2020-04-01	631.583253	1254.538710	0.000000	-622.955457	11.71	220.0	260.0	1300	1100
3	2020-04-01	642.718380	1127.648368	0.000000	-484.929988	11.71	220.0	260.0	1300	1100
4	2020-04-01	1207.234188	1142.545229	64.688959	0.000000	11.71	220.0	260.0	1300	1100
5	2020-04-01	849.707278	1300.000000	0.000000	-450.292722	11.71	220.0	260.0	1300	1100
6	2020-04-01	550.462581	799.488549	0.000000	-249.025969	11.71	220.0	260.0	1300	1100
7	2020-04-01	814.677043	1156.212769	0.000000	-341.535726	11.71	220.0	260.0	1300	1100
8	2020-04-01	953.731351	925.692632	28.038719	0.000000	11.71	220.0	260.0	1300	1100
9	2020-04-01	576.525796	1300.000000	0.000000	-723.474204	11.71	220.0	260.0	1300	1100
10	2020-04-01	644.638855	1051.988194	0.000000	-407.349338	11.71	220.0	260.0	1300	1100
1	2020-05-01	912.760967	1300.000000	0.000000	-387.239033	9.84	220.0	260.0	1300	1100
2	2020-05-01	739.712334	1300.000000	0.000000	-560.287666	9.84	220.0	260.0	1300	1100
3	2020-05-01	775.732607	1077.631521	0.000000	-301.898914	9.84	220.0	260.0	1300	1100
4	2020-05-01	715.108925	1300.000000	0.000000	-584.891075	9.84	220.0	260.0	1300	1100
5	2020-05-01	876.251881	1159.808066	0.000000	-283.556185	9.84	220.0	260.0	1300	1100
6	2020-05-01	816.994069	1227.743378	0.000000	-410.749309	9.84	220.0	260.0	1300	1100
7	2020-05-01	873.731691	1299.865021	0.000000	-426.133330	9.84	220.0	260.0	1300	1100

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean
8	2020-05-01	503.021401	721.002799	0.000000	-217.981398	9.84	220.0	260.0	1300	1100
9	2020-05-01	677.085433	1300.000000	0.000000	-622.914567	9.84	220.0	260.0	1300	1100
10	2020-05-01	774.918096	1300.000000	0.000000	-525.081904	9.84	220.0	260.0	1300	1100
1	2020-06-01	1038.681500	1020.879060	17.802440	0.000000	11.53	220.0	260.0	1300	1100
2	2020-06-01	842.573368	1300.000000	0.000000	-457.426632	11.53	220.0	260.0	1300	1100
3	2020-06-01	962.018220	910.318008	51.700212	0.000000	11.53	220.0	260.0	1300	1100
4	2020-06-01	1033.504748	1300.000000	0.000000	-266.495252	11.53	220.0	260.0	1300	1100
5	2020-06-01	1004.271183	1300.000000	0.000000	-295.728817	11.53	220.0	260.0	1300	1100
6	2020-06-01	910.162557	948.475309	0.000000	-38.312753	11.53	220.0	260.0	1300	1100
7	2020-06-01	995.628429	873.371255	122.257174	0.000000	11.53	220.0	260.0	1300	1100
8	2020-06-01	727.989659	1185.416210	0.000000	-457.426551	11.53	220.0	260.0	1300	1100
9	2020-06-01	1014.573140	1212.802337	0.000000	-198.229197	11.53	220.0	260.0	1300	1100
10	2020-06-01	931.843701	1136.837981	0.000000	-204.994280	11.53	220.0	260.0	1300	1100
1	2020-07-01	1165.090456	1300.000000	0.000000	-134.909544	11.71	220.0	260.0	1300	1100
2	2020-07-01	772.752521	1300.000000	0.000000	-527.247479	11.71	220.0	260.0	1300	1100
3	2020-07-01	735.696415	1300.000000	0.000000	-564.303585	11.71	220.0	260.0	1300	1100
4	2020-07-01	1017.990935	1025.455744	0.000000	-7.464809	11.71	220.0	260.0	1300	1100

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean
5	2020-07-01	1117.175491	1300.000000	0.000000	-182.824509	11.71	220.0	260.0	1300	1100
6	2020-07-01	1185.948752	1300.000000	0.000000	-114.051248	11.71	220.0	260.0	1300	1100
7	2020-07-01	1082.666641	1263.519159	0.000000	-180.852518	11.71	220.0	260.0	1300	1100
8	2020-07-01	1051.584425	1300.000000	0.000000	-248.415575	11.71	220.0	260.0	1300	1100
9	2020-07-01	849.363106	1300.000000	0.000000	-450.636894	11.71	220.0	260.0	1300	1100
10	2020-07-01	1390.954298	1232.187925	158.766373	0.000000	11.71	220.0	260.0	1300	1100
1	2020-08-01	1236.711727	1300.000000	0.000000	-63.288273	11.61	220.0	260.0	1300	1100
2	2020-08-01	1320.799390	975.817933	344.981457	0.000000	11.61	220.0	260.0	1300	1100
3	2020-08-01	1162.721526	1013.844404	148.877122	0.000000	11.61	220.0	260.0	1300	1100
4	2020-08-01	738.076595	1300.000000	0.000000	-561.923405	11.61	220.0	260.0	1300	1100
5	2020-08-01	1261.774214	1107.427719	154.346494	0.000000	11.61	220.0	260.0	1300	1100
6	2020-08-01	990.265411	1300.000000	0.000000	-309.734589	11.61	220.0	260.0	1300	1100
7	2020-08-01	1378.035985	1264.077101	113.958884	0.000000	11.61	220.0	260.0	1300	1100
8	2020-08-01	1289.387387	1300.000000	0.000000	-10.612613	11.61	220.0	260.0	1300	1100
9	2020-08-01	1136.180265	1300.000000	0.000000	-163.819735	11.61	220.0	260.0	1300	1100
10	2020-08-01	1245.287440	1047.405482	197.881958	0.000000	11.61	220.0	260.0	1300	1100
1	2020-09-01	891.556610	1298.843324	0.000000	-407.286714	11.97	220.0	260.0	1300	1100

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean
2	2020-09-01	455.297268	1300.000000	0.000000	-844.702732	11.97	220.0	260.0	1300	1100
3	2020-09-01	796.070701	1098.566634	0.000000	-302.495932	11.97	220.0	260.0	1300	1100
4	2020-09-01	860.703025	1297.552831	0.000000	-436.849806	11.97	220.0	260.0	1300	1100
5	2020-09-01	1112.831083	1109.574168	3.256915	0.000000	11.97	220.0	260.0	1300	1100
6	2020-09-01	878.165805	979.311590	0.000000	-101.145784	11.97	220.0	260.0	1300	1100
7	2020-09-01	943.177991	876.288588	66.889403	0.000000	11.97	220.0	260.0	1300	1100
8	2020-09-01	929.591289	884.121330	45.469959	0.000000	11.97	220.0	260.0	1300	1100
9	2020-09-01	1144.838420	1300.000000	0.000000	-155.161580	11.97	220.0	260.0	1300	1100
10	2020-09-01	773.464882	1300.000000	0.000000	-526.535118	11.97	220.0	260.0	1300	1100
1	2020-10-01	1130.832481	783.993174	346.839307	0.000000	11.71	220.0	260.0	1300	1100
2	2020-10-01	1235.486678	1300.000000	0.000000	-64.513322	11.71	220.0	260.0	1300	1100
3	2020-10-01	859.968470	1300.000000	0.000000	-440.031530	11.71	220.0	260.0	1300	1100
4	2020-10-01	805.362242	1252.774736	0.000000	-447.412494	11.71	220.0	260.0	1300	1100
5	2020-10-01	313.363713	744.759198	0.000000	-431.395485	11.71	220.0	260.0	1300	1100
6	2020-10-01	604.781916	978.225231	0.000000	-373.443315	11.71	220.0	260.0	1300	1100
7	2020-10-01	792.522779	1300.000000	0.000000	-507.477221	11.71	220.0	260.0	1300	1100
8	2020-10-01	1070.995679	1066.535551	4.460128	0.000000	11.71	220.0	260.0	1300	1100

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean
9	2020-10-01	1041.652274	1020.652273	21.000001	0.000000	11.71	220.0	260.0	1300	1100
10	2020-10-01	1088.068193	1061.982595	26.085598	0.000000	11.71	220.0	260.0	1300	1100
1	2020-11-01	806.965579	1103.241179	0.000000	-296.275600	12.00	220.0	260.0	1300	1100
2	2020-11-01	790.826705	979.045000	0.000000	-188.218295	12.00	220.0	260.0	1300	1100
3	2020-11-01	722.210544	1152.441576	0.000000	-430.231032	12.00	220.0	260.0	1300	1100
4	2020-11-01	779.390564	1012.187101	0.000000	-232.796536	12.00	220.0	260.0	1300	1100
5	2020-11-01	883.079338	1033.644700	0.000000	-150.565363	12.00	220.0	260.0	1300	1100
6	2020-11-01	748.715813	815.914776	0.000000	-67.198963	12.00	220.0	260.0	1300	1100
7	2020-11-01	769.091922	1153.608496	0.000000	-384.516574	12.00	220.0	260.0	1300	1100
8	2020-11-01	837.666968	739.516813	98.150154	0.000000	12.00	220.0	260.0	1300	1100
9	2020-11-01	804.379282	1300.000000	0.000000	-495.620718	12.00	220.0	260.0	1300	1100
10	2020-11-01	850.428711	844.553833	5.874878	0.000000	12.00	220.0	260.0	1300	1100
1	2020-12-01	788.236454	1300.000000	0.000000	-511.763546	11.86	220.0	260.0	1300	1100
2	2020-12-01	750.823892	463.303099	287.520793	0.000000	11.86	220.0	260.0	1300	1100
3	2020-12-01	293.502601	1013.254735	0.000000	-719.752134	11.86	220.0	260.0	1300	1100
4	2020-12-01	484.845007	1300.000000	0.000000	-815.154993	11.86	220.0	260.0	1300	1100
5	2020-12-01	601.377546	1133.709775	0.000000	-532.332230	11.86	220.0	260.0	1300	1100

id	time	demand	generation	consumption	net_energy	price	demand_std	generation_std	generation_mean	demand_mean
6	2020-12-01	524.675582	947.526710	0.000000	-422.851128	11.86	220.0	260.0	1300	1100
7	2020-12-01	756.070294	1300.000000	0.000000	-543.929706	11.86	220.0	260.0	1300	1100
8	2020-12-01	811.233796	1286.337795	0.000000	-475.103999	11.86	220.0	260.0	1300	1100
9	2020-12-01	698.712910	1186.312547	0.000000	-487.599638	11.86	220.0	260.0	1300	1100
10	2020-12-01	589.634455	1202.460345	0.000000	-612.825891	11.86	220.0	260.0	1300	1100

Conclusion

The payments for NRG before a coalition and after a colition where shown in the printed tables for N=3,5,10 prosumers. The results show that there was not a significant change between these methods that could be seen statistically. The distribution seems to already include the division of contirbution equally for each prosumer. Other methods to provide coalitional characteristic values can be explored to leverage the non-linear payment of the consumers on the network.

In [ ]: