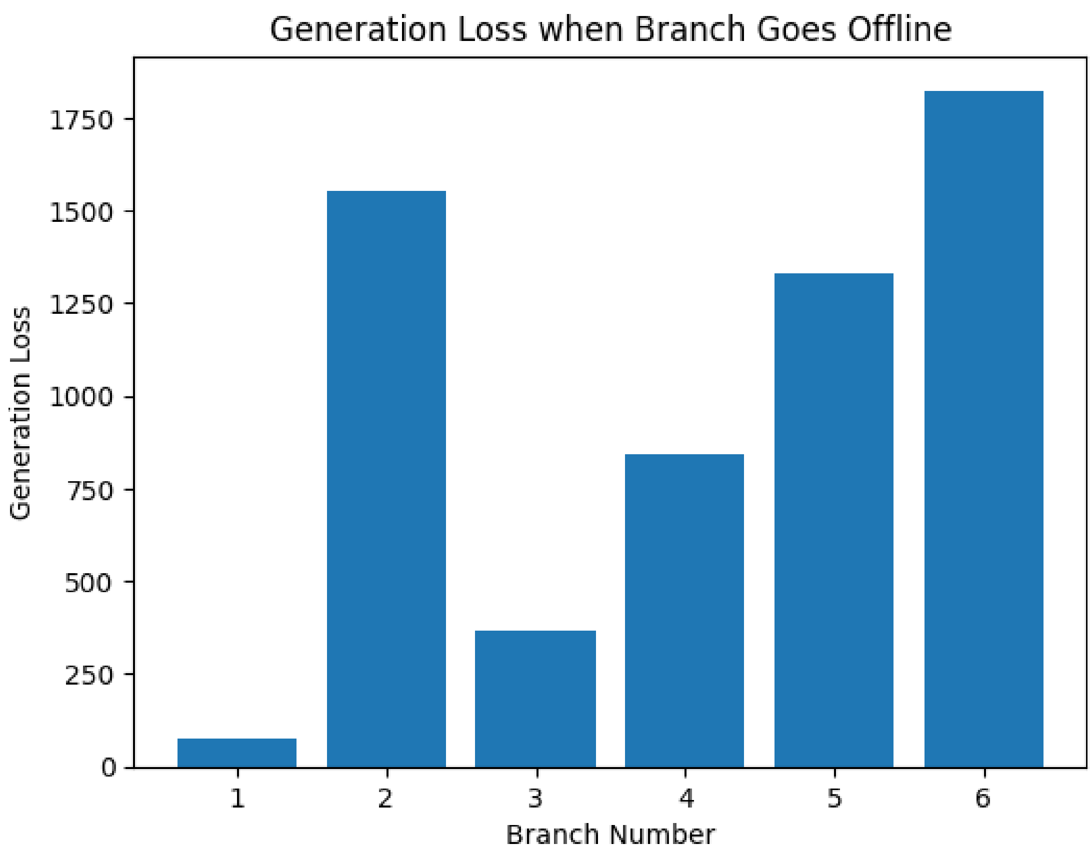
ex

November 30, 2023

[456]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import math
import itertools
import seaborn
```

[457]:
```
#Compute Generation Losses on Loss of Branches
n=6 #set number of branches here
gen_loss=np.random.randint(low=30,high=2000,size=(1,n))
gen_loss=gen_loss[0]
print(gen_loss)
list1=[]
for i in range(1,len(gen_loss)+1):
    list1.append(i)
print(list1)
namelist=[]
for i in list1:
    namelist.append(str(i))
plt.bar(namelist,gen_loss)
plt.title('Generation Loss when Branch Goes Offline')
plt.xlabel('Branch Number')
plt.ylabel('Generation Loss')
plt.show()
```

```
[  77 1555  369  844 1332 1824]
[1, 2, 3, 4, 5, 6]
```

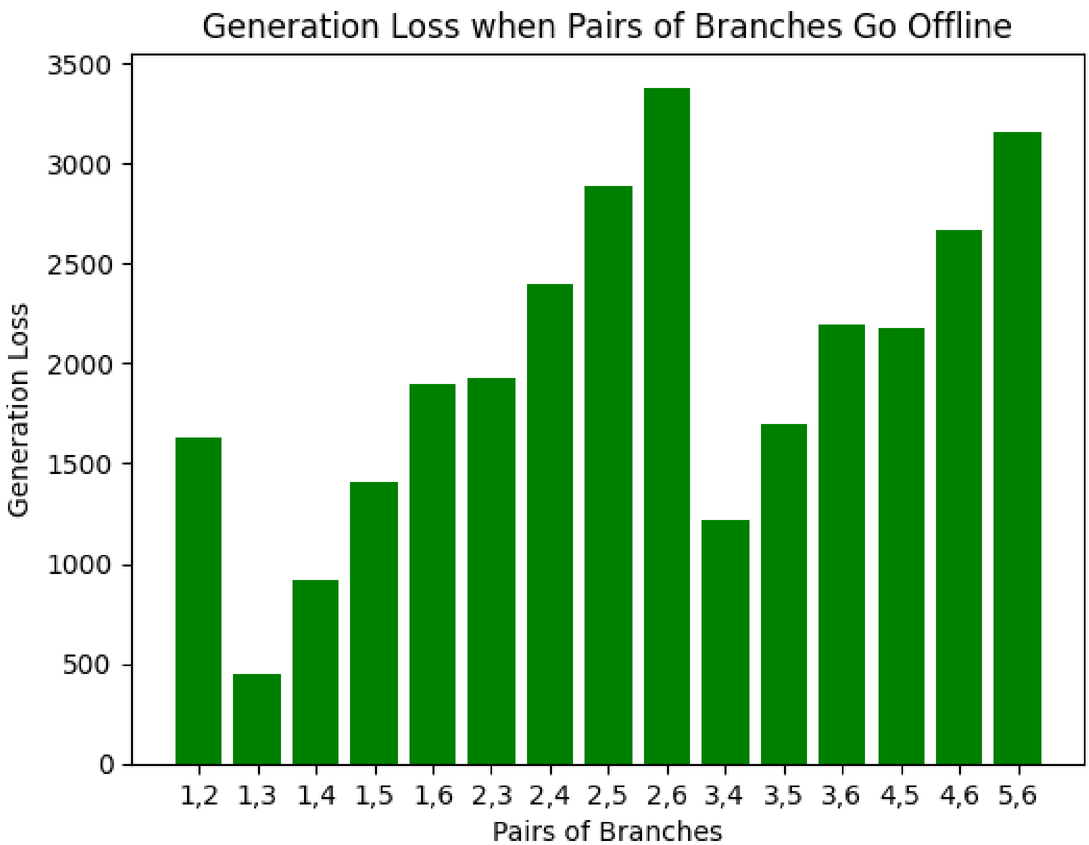


[458]:
```
a=math.comb(n,2)
print(a)
gen_pair_loss=np.zeros((a),dtype='i')
print(gen_pair_loss)
list_of_tuples=[]
list_of_indices=[]
i=0
for comb in itertools.combinations(list1, 2):
    list_of_tuples.append(comb)
    r=str(comb[0])+','+str(comb[1])
    list_of_indices.append(r)
    gen_pair_loss[i]=gen_loss[comb[0]-1]+gen_loss[comb[1]-1]
    i+=1
print(gen_pair_loss)
plt.bar(list_of_indices,gen_pair_loss,color='green')
plt.xlabel("Pairs of Branches")
plt.ylabel("Generation Loss")
plt.title("Generation Loss when Pairs of Branches Go Offline")
```

15

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[1632  446  921 1409 1901 1924 2399 2887 3379 1213 1701 2193 2176 2668
 3156]
```

[458]: Text(0.5, 1.0, 'Generation Loss when Pairs of Branches Go Offline')



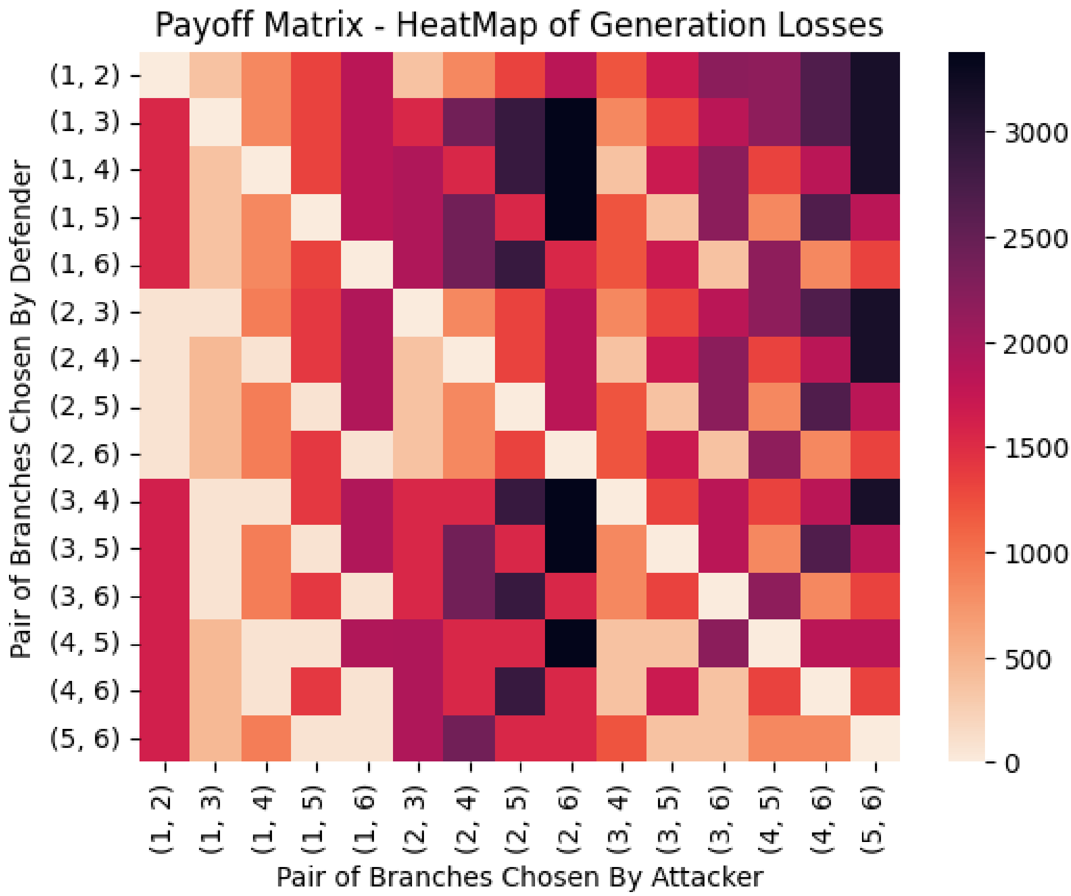
[459]:
```
payoff=np.zeros((a,a),dtype='i')
for i in range(0,a):
    for j in range(0,a):
        q=gen_pair_loss[j]
        set1={list_of_tuples[j][0],list_of_tuples[j][1]}
        set2={list_of_tuples[i][0],list_of_tuples[i][1]}
        set1=set1.intersection(set2)
        if len(set1)>0:
            for k in set1:
                q=q-gen_loss[k-1]
        payoff[i][j]=q
print(payoff)
```

```
[[   0  369  844 1332 1824  369  844 1332 1824 1213 1701 2193 2176 2668
  3156]
```

```
 [1555    0  844 1332 1824 1555 2399 2887 3379  844 1332 1824 2176 2668
  3156]
 [1555  369    0 1332 1824 1924 1555 2887 3379  369 1701 2193 1332 1824
  3156]
 [1555  369  844    0 1824 1924 2399 1555 3379 1213  369 2193  844 2668
  1824]
 [1555  369  844 1332    0 1924 2399 2887 1555 1213 1701  369 2176  844
  1332]
 [  77   77  921 1409 1901    0  844 1332 1824  844 1332 1824 2176 2668
  3156]
 [  77  446   77 1409 1901  369    0 1332 1824  369 1701 2193 1332 1824
  3156]
 [  77  446  921   77 1901  369  844    0 1824 1213  369 2193  844 2668
  1824]
 [  77  446  921 1409   77  369  844 1332    0 1213 1701  369 2176  844
  1332]
 [1632   77   77 1409 1901 1555 1555 2887 3379    0 1332 1824 1332 1824
  3156]
 [1632   77  921   77 1901 1555 2399 1555 3379  844    0 1824  844 2668
  1824]
 [1632   77  921 1409   77 1555 2399 2887 1555  844 1332    0 2176  844
  1332]
 [1632  446   77   77 1901 1924 1555 1555 3379  369  369 2193    0 1824
  1824]
 [1632  446   77 1409   77 1924 1555 2887 1555  369 1701  369 1332    0
  1332]
 [1632  446  921   77   77 1924 2399 1555 1555 1213  369  369  844  844
     0]]
```

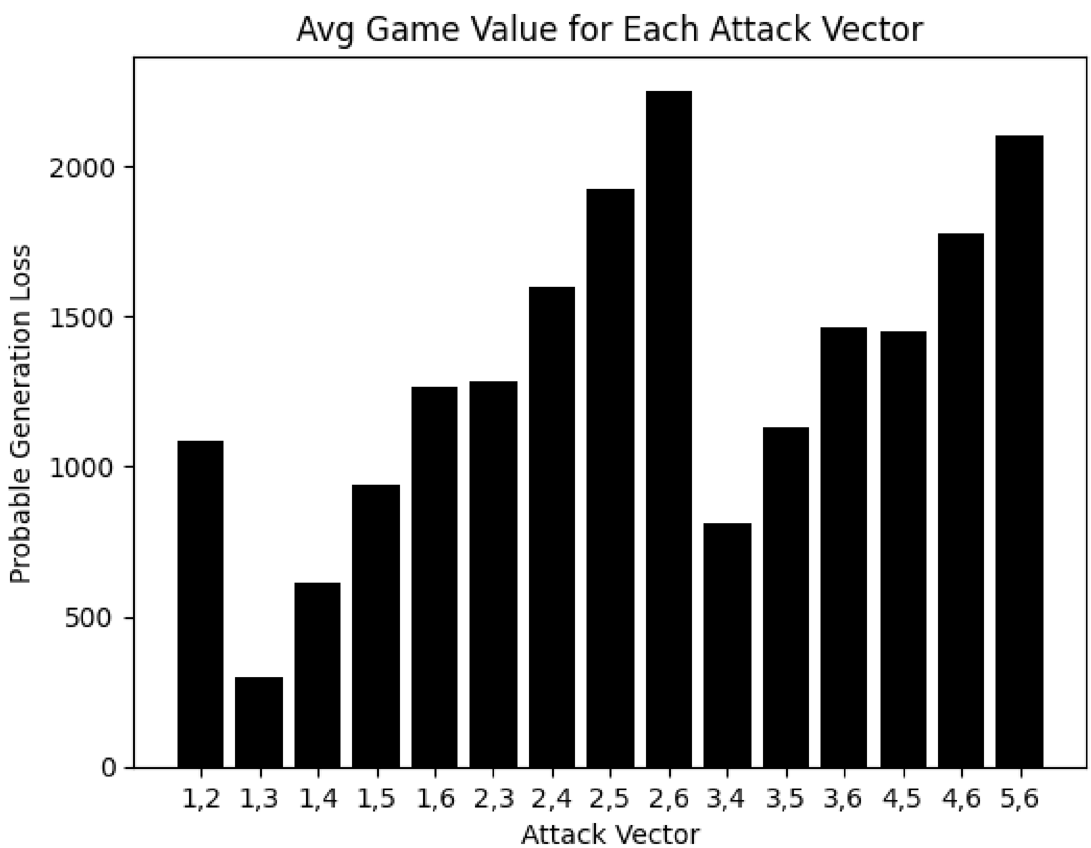
[469]:

```
payoff_map=pd.
 ↪DataFrame(payoff,columns=list_of_tuples,index=list_of_tuples,dtype='f')
seaborn.heatmap(payoff_map,cmap=seaborn.cm.rocket_r)
plt.title("Payoff Matrix - HeatMap of Generation Losses")
plt.xlabel("Pair of Branches Chosen By Attacker")
plt.ylabel("Pair of Branches Chosen By Defender")
plt.show()
```



[476]:

```
#Case-1: If Defender was static::defender choses each defence vector pair with
↪equal probability
#then our attacker must chose the column of payoff matrix that has highest sum
↪of values
game_value=payoff_map.sum(axis=0).values
for i in range(0,len(game_value)):
    game_value[i]/=math.comb(n,2)
plt.xlabel("Attack Vector")
plt.ylabel("Probable Generation Loss")
plt.title("Avg Game Value for Each Attack Vector")
plt.bar(list_of_indices,game_value,color='#000000')
plt.show()
```



[477]:
```
#Case-1 solution
print("Best Attack Vector in Case-1 is: ",payoff_map.sum(axis=0).idxmax())
print("Best Probable Generation Loss is: ",round((payoff_map.
    ↪sum(axis=0)[payoff_map.sum(axis=0).idxmax()])/math.comb(n,2)))
```

```
Best Attack Vector in Case-1 is:  (2, 6)
Best Probable Generation Loss is:  2253
```

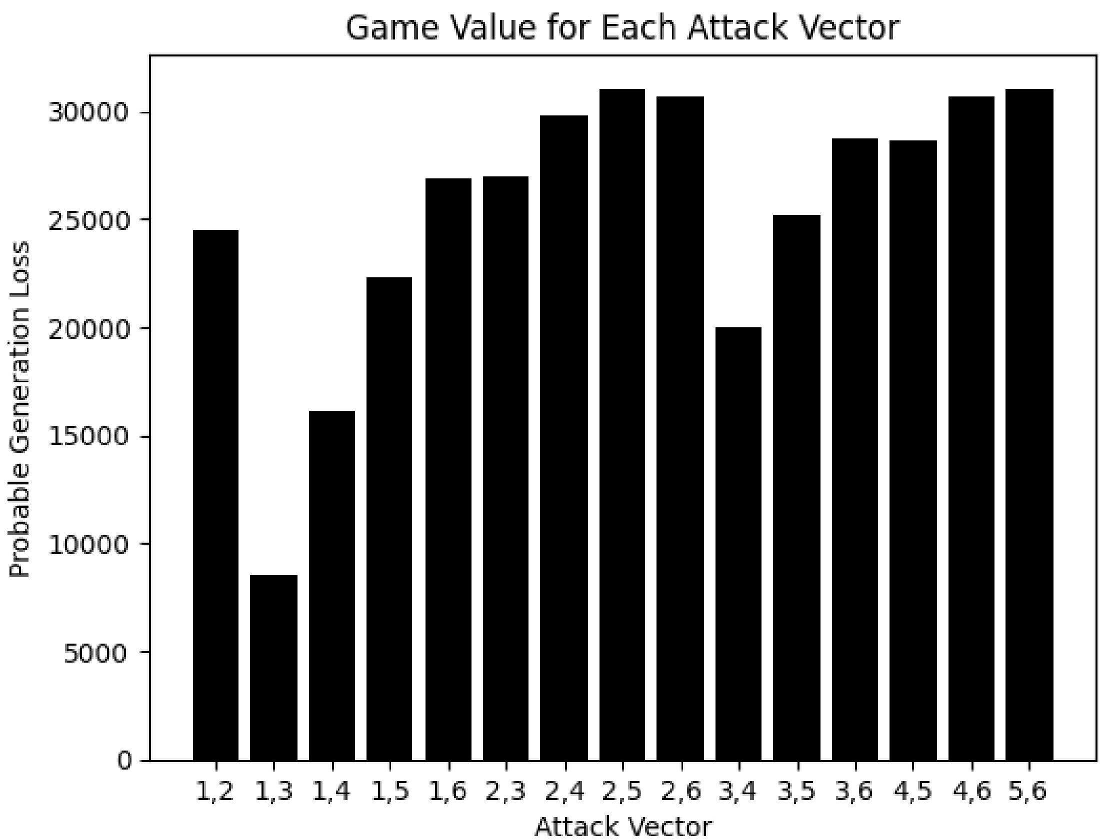
[478]:
```
#Case-2 Defender is intelligent
#Here defender chooses defence vector to protect his more valuable branch pairs␣
    ↪with a higher frequency.
probability_vector=[]
sum=gen_pair_loss.sum()
for i in range(0,len(gen_pair_loss)):
    probability_vector.append(1-((gen_pair_loss[i])/sum))
p_vector=np.array(probability_vector,dtype='f')
p_vector = (p_vector - np.min(p_vector)+np.min(p_vector)*0.1)/(np.
    ↪max(p_vector)-np.min(p_vector))
print(p_vector)
```

```
[1.5034441 1.9078078 1.7458576 1.5794749 1.4117289 1.4038874 1.241937
```

```
1.0755545 0.9078078 1.6463008 1.4799182 1.3121722 1.3179679 1.150222
0.9838393]
```

[479]:
```
#for every attack-vector column, the probable generation-loss is now the mean␣
↪of modified values. The modified values refer to
#the rows of payoff matrix multilplied by corresponding probability factor␣
↪p_vector.
modified_payoff=payoff_map.mul(p_vector,axis=1)
```

[480]:
```
gr=modified_payoff.sum(axis=0).values
plt.xlabel("Attack Vector")
plt.ylabel("Probable Generation Loss")
plt.title("Game Value for Each Attack Vector")
plt.bar(list_of_indices,gr,color='#000000')
plt.show()
```



[481]:
```
#Case-2 solution
print("Best Attack Vector in Case-2 is: ",modified_payoff.sum(axis=0).idxmax())
print("Best Probable Generation Loss is: ",round((payoff_map.
↪sum(axis=0)[modified_payoff.sum(axis=0).idxmax()])/math.comb(n,2)))
```

```
Best Attack Vector in Case-2 is:  (2, 5)
Best Probable Generation Loss is:  1925
```

[ ]:

```

```