In [1]:
```
import numpy as np
import pandas as pd
```

Helper Functions

In [2]:
```
def output_data(file):
    data = pd.read_csv(file, sep=" ", header=None)
    data.columns = ["k", "s", "cost", "file", "algorithm"]
    return data

def tsp_compare(graph_size, tsp_iterations):
    tsp_costs = {}
    for current in tsp_iterations:
        current_data = output_data('best_tsp_' + str(graph_size) + '_iter'
        for comparison in tsp_iterations:
            compare_data = output_data('best_tsp_' + str(graph_size) + '_it
            if current != comparison:
                num_same = sum(current_data['cost'] == compare_data['cost']
                if current not in tsp_costs:
                    tsp_costs[current] = [num_same]
                else:
                    tsp_costs[current].append(num_same)
    return tsp_costs

def compare_tsp_and_ant_50():
    tsp_iterations = range(100, 1100, 100)
    costs = {}
    ant_data = output_data('best_ant_50.txt')
    for current in tsp_iterations:
        current_data = output_data('best_tsp_50_iter' + str(current)  + '.t
        num_same = sum(current_data['cost'] == ant_data['cost'])
        if current not in costs:
            costs[current] = [num_same]
        else:
            costs[current].append(num_same)
    return costs
```

Comparing Iterations 100 to 1000 for Size 50

Surprisingly, there is no difference between each iteration. Costs are exactly the same.

In [3]:
```
tsp_compare(50, range(100, 1100, 100))
```

Out[3]:
```
{100: [325, 325, 325, 325, 325, 325, 325, 325, 325],
 200: [325, 325, 325, 325, 325, 325, 325, 325, 325],
 300: [325, 325, 325, 325, 325, 325, 325, 325, 325],
 400: [325, 325, 325, 325, 325, 325, 325, 325, 325],
 500: [325, 325, 325, 325, 325, 325, 325, 325, 325],
 600: [325, 325, 325, 325, 325, 325, 325, 325, 325],
 700: [325, 325, 325, 325, 325, 325, 325, 325, 325],
 800: [325, 325, 325, 325, 325, 325, 325, 325, 325],
 900: [325, 325, 325, 325, 325, 325, 325, 325, 325],
 1000: [325, 325, 325, 325, 325, 325, 325, 325, 325]}
```

In [4]:
```
compare_tsp_and_ant_50()
```

Out[4]:
```
{100: [74],
 200: [74],
 300: [74],
 400: [74],
 500: [74],
 600: [74],
 700: [74],
 800: [74],
 900: [74],
 1000: [74]}
```

Comparing Iterations 100 to 1000 for Size 100

Surprisingly, there is no difference between each iteration. Costs are exactly the same.

In [5]:
```
tsp_compare(100, range(100, 1100, 100))
```

Out[5]:
```
{100: [314, 314, 314, 314, 314, 314, 314, 314, 314],
 200: [314, 314, 314, 314, 314, 314, 314, 314, 314],
 300: [314, 314, 314, 314, 314, 314, 314, 314, 314],
 400: [314, 314, 314, 314, 314, 314, 314, 314, 314],
 500: [314, 314, 314, 314, 314, 314, 314, 314, 314],
 600: [314, 314, 314, 314, 314, 314, 314, 314, 314],
 700: [314, 314, 314, 314, 314, 314, 314, 314, 314],
 800: [314, 314, 314, 314, 314, 314, 314, 314, 314],
 900: [314, 314, 314, 314, 314, 314, 314, 314, 314],
 1000: [314, 314, 314, 314, 314, 314, 314, 314, 314]}
```

Comparing Ant Colony and 2-opt

In [6]:
```
ant_data = output_data('best_ant_50.txt')
ant_data['algorithm'] = 'ant'
ant_data.head()
```

Out[6]:

	k	s	cost	file	algorithm
0	24	19	2.106994e+02	inputs/85_50.in	ant
1	26	19	3.769865e+03	inputs/192_50.in	ant
2	49	19	4.333333e+00	inputs/293_50.in	ant
3	24	19	7.858472e+01	inputs/78_50.in	ant
4	25	19	1.208264e+09	inputs/152_50.in	ant

In [7]:
```
tsp_data = output_data('best_tsp_50_iter100.txt')
tsp_data['algorithm'] = 'tsp'
tsp_data.head()
```

Out[7]:

	k	s	cost	file	algorithm
0	1	1	2.019085e+02	inputs/85_50.in	tsp
1	1	1	3.770212e+03	inputs/192_50.in	tsp
2	1	1	5.333333e+00	inputs/293_50.in	tsp
3	1	1	7.277832e+01	inputs/78_50.in	tsp
4	1	1	1.159118e+09	inputs/152_50.in	tsp

In [8]:
```
merged_tsp_ant = pd.merge(ant_data, tsp_data, left_on='file', right_on='fil
merged_tsp_ant.head()
```

Out[8]:

	k_ant	s_ant	cost_ant	file	algorithm_ant	k_tsp	s_tsp	cost_tsp	algorithm
0	24	19	2.106994e+02	inputs/85_50.in	ant	1	1	2.019085e+02	
1	26	19	3.769865e+03	inputs/192_50.in	ant	1	1	3.770212e+03	
2	49	19	4.333333e+00	inputs/293_50.in	ant	1	1	5.333333e+00	
3	24	19	7.858472e+01	inputs/78_50.in	ant	1	1	7.277832e+01	
4	25	19	1.208264e+09	inputs/152_50.in	ant	1	1	1.159118e+09	

In [9]:
```
ant_better = merged_tsp_ant[merged_tsp_ant['cost_ant'] < merged_tsp_ant['co
print(ant_better.shape)
ant_better.head()
```

```
(58, 9)
```

Out[9]:

	k_ant	s_ant	cost_ant	file	algorithm_ant	k_tsp	s_tsp	cost_tsp	algori
1	26	19	3769.865333	inputs/192_50.in	ant	1	1	3770.211560	
2	49	19	4.333333	inputs/293_50.in	ant	1	1	5.333333	
8	22	19	203.125650	inputs/216_50.in	ant	1	1	209.623167	
10	25	19	32.000000	inputs/232_50.in	ant	10	7	32.333333	
16	28	19	100401.333333	inputs/248_50.in	ant	1	1	101120.666667	

In [10]:
```
tsp_better = merged_tsp_ant[merged_tsp_ant['cost_ant'] >= merged_tsp_ant['c
print(tsp_better.shape)
tsp_better.head()
```

```
(267, 9)
```

Out[10]:

	k_ant	s_ant	cost_ant	file	algorithm_ant	k_tsp	s_tsp	cost_tsp	algorithm
0	24	19	2.106994e+02	inputs/85_50.in	ant	1	1	2.019085e+02	
3	24	19	7.858472e+01	inputs/78_50.in	ant	1	1	7.277832e+01	
4	25	19	1.208264e+09	inputs/152_50.in	ant	1	1	1.159118e+09	
5	21	19	2.080000e+02	inputs/117_50.in	ant	1	1	2.080000e+02	
6	27	19	4.544703e+02	inputs/45_50.in	ant	11	7	4.493073e+02	

Conclusion

2-opt does better than Ant Colony on average.

In [ ]:
```
```