AIND Planning: heuristic analysis

For the Air Cargo Planning problem, we attempt to solve three problems with our planning solutions:

- Problem 1 initial state and goal:

```
Init(At(C1, SFO) ∧ At(C2, JFK)
  ∧ At(P1, SFO) ∧ At(P2, JFK)
  ∧ Cargo(C1) ∧ Cargo(C2)
  ∧ Plane(P1) ∧ Plane(P2)
  ∧ Airport(JFK) ∧ Airport(SFO))
Goal(At(C1, JFK) ∧ At(C2, SFO))
```

- Problem 2 initial state and goal:

```
Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL)
  ∧ At(P1, SFO) ∧ At(P2, JFK) ∧ At(P3, ATL)
  ∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3)
  ∧ Plane(P1) ∧ Plane(P2) ∧ Plane(P3)
  ∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL))
Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, SFO))
```

- Problem 3 initial state and goal:

```
Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL) ∧ At(C4, ORD)
  ∧ At(P1, SFO) ∧ At(P2, JFK)
  ∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3) ∧ Cargo(C4)
  ∧ Plane(P1) ∧ Plane(P2)
  ∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL) ∧ Airport(ORD))
Goal(At(C1, JFK) ∧ At(C3, JFK) ∧ At(C2, SFO) ∧ At(C4, SFO))
```

An optimal plan for Problem 1:

```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```

An optimal plan for Problem 2:

```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, ATL)
Unload(C1, P1, JFK)
Load(C3, P2, ATL)
Fly(P2, ATL, SFO)
Unload(C3, P2, SFO)
Unload(C2, P2, SFO)
```

An optimal plan for Problem 3:

```
Load(C2, P2, JFK)
Load(C1, P1, SFO)
Fly(P2, JFK, ORD)
Fly(P1, SFO, ATL)
Load(C4, P2, ORD)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)
Unload(C3, P1, JFK)
Fly(P2, ORD, SFO)
Unload(C2, P2, SFO)
Unload(C4, P2, SFO)
```

Data review

Below we will take a look at the collected data and review each problem and search time results. Results from each problem/search were preserved in a CSV and graphs etc created below

In [10]:
```
import os,sys,time,random,math,time
import tarfile, zipfile

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import matplotlib.pyplot as plt
from IPython.display import display, Image
%matplotlib inline

def loadData(datadir,filename):
    # Load the wholesale customers dataset
    #data = pd.read_csv(filename)
    data = ''
    print ("loading: "+datadir+filename)
    try:
        if zipfile.is_zipfile(datadir+filename):
            z = zipfile.ZipFile(datadir+filename)
            filename = z.open(filename[:-4])
        else:
            filename=datadir+filename
        data = pd.read_csv(filename, parse_dates=True)
        print ("Dataset has {} samples with {} features each.".format(*data.shape))
    except Exception as e:
        print ("Dataset could not be loaded. Is the dataset missing?")
        print(e)
    return data

# load our data
datadir='./'
data = loadData(datadir,'run_search_results.csv')
display(data.info())


# convert the Actions list to something more useful, as well as add a count of actions field
```

```
actions=[]
num_actions=[]
for each in data['Actions']:
    actions.append(each[1:-1][1:-1].split("\', \'"))

for each in actions:
    num_actions.append(len(each))
data['Actions']=actions
data['Num_Actions']=num_actions
display(data)
```

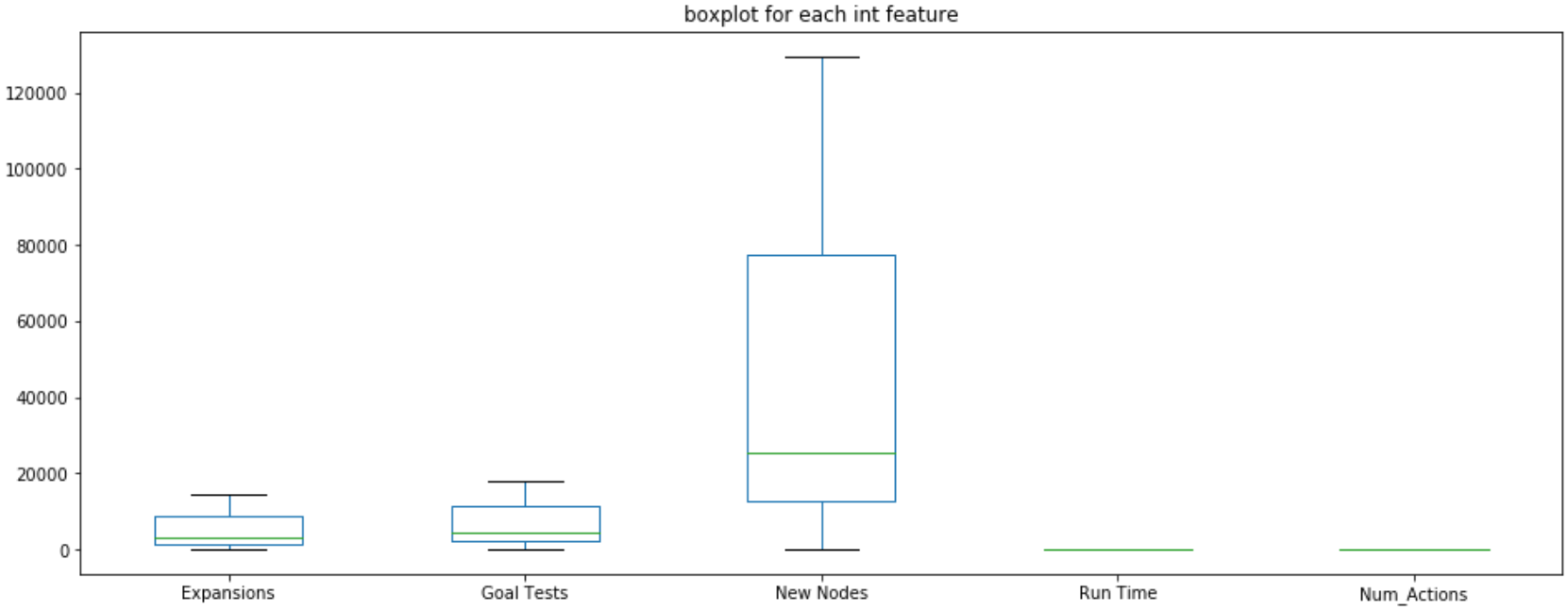
```
loading: ./run_search_results.csv
Dataset has 21 samples with 7 features each.
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 7 columns):
curr_prob      21 non-null object
curr_search    21 non-null object
Expansions     21 non-null int64
Goal Tests     21 non-null int64
New Nodes      21 non-null int64
Run Time       21 non-null float64
Actions        21 non-null object
dtypes: float64(1), int64(3), object(3)
memory usage: 1.2+ KB

None
```

	curr_prob	curr_search	Expansions	Goal Tests	New Nodes	Run Time	Actions	Num_Actions
0	Air Cargo Problem 1	breadth_first_search	43	56	180	0.027285	[Load(C2, P2, JFK), Load(C1, P1, SFO), Fly(P2,...	6
1	Air Cargo Problem 1	depth_first_graph_search	12	13	48	0.008828	[Fly(P1, SFO, JFK), Fly(P2, JFK, SFO), Load(C1...	12
2	Air Cargo Problem 1	uniform_cost_search	55	57	224	0.036490	[Load(C1, P1, SFO), Load(C2, P2, JFK), Fly(P1,...	6
3	Air Cargo Problem 1	greedy_best_first_graph_search with h_1	7	9	28	0.003639	[Load(C1, P1, SFO), Load(C2, P2, JFK), Fly(P1,...	6
4	Air Cargo Problem 1	astar_search with h_1	55	57	224	0.032251	[Load(C1, P1, SFO), Load(C2, P2, JFK), Fly(P1,...	6
5	Air Cargo Problem 1	astar_search with h_ignore_preconditions	41	43	170	0.030720	[Load(C1, P1, SFO), Fly(P1, SFO, JFK), Unload(...	6
6	Air Cargo Problem 1	astar_search with h_pg_levelsum	11	13	50	0.540841	[Load(C1, P1, SFO), Fly(P1, SFO, JFK), Load(C2...	6
7	Air Cargo Problem 2	breadth_first_search	3063	4274	25442	8.716775	[Load(C2, P2, JFK), Load(C1, P1, SFO), Fly(P2,...	9
8	Air Cargo Problem 2	depth_first_graph_search	42	43	260	0.069772	[Fly(P3, ATL, SFO), Fly(P1, SFO, ATL), Fly(P2,...	37

	curr_prob	curr_search	Expansions	Goal Tests	New Nodes	Run Time	Actions	Num_Actions
9	Air Cargo Problem 2	uniform_cost_search	4395	4397	36060	8.175415	[Load(C1, P1, SFO), Load(C2, P2, JFK), Fly(P2,...	9
10	Air Cargo Problem 2	greedy_best_first_graph_search with h_1	1353	1355	10660	2.541624	[Load(C1, P1, SFO), Load(C2, P2, JFK), Load(C3...	27
11	Air Cargo Problem 2	astar_search with h_1	4395	4397	36060	8.816888	[Load(C1, P1, SFO), Load(C2, P2, JFK), Fly(P2,...	9
12	Air Cargo Problem 2	astar_search with h_ignore_preconditions	1294	1296	10927	3.170810	[Load(C2, P2, JFK), Fly(P2, JFK, ATL), Load(C3...	9
13	Air Cargo Problem 2	astar_search with h_pg_levelsum	252	254	2055	84.355572	[Load(C2, P2, JFK), Fly(P2, JFK, ATL), Load(C3...	9
14	Air Cargo Problem 3	breadth_first_search	14663	18098	129631	92.402491	[Load(C2, P2, JFK), Load(C1, P1, SFO), Fly(P2,...	12
15	Air Cargo Problem 3	depth_first_graph_search	592	593	4927	2.591931	[Fly(P1, SFO, ORD), Fly(P2, JFK, ORD), Fly(P1,...	571
16	Air Cargo Problem 3	uniform_cost_search	18235	18237	159716	45.120041	[Load(C1, P1, SFO), Load(C2, P2, JFK), Fly(P1,...	12
17	Air Cargo Problem 3	greedy_best_first_graph_search with h_1	5614	5616	49429	13.997447	[Load(C1, P1, SFO), Load(C2, P2, JFK), Fly(P1,...	22
18	Air Cargo Problem 3	astar_search with h_1	18235	18237	159716	45.839645	[Load(C1, P1, SFO), Load(C2, P2, JFK), Fly(P1,...	12

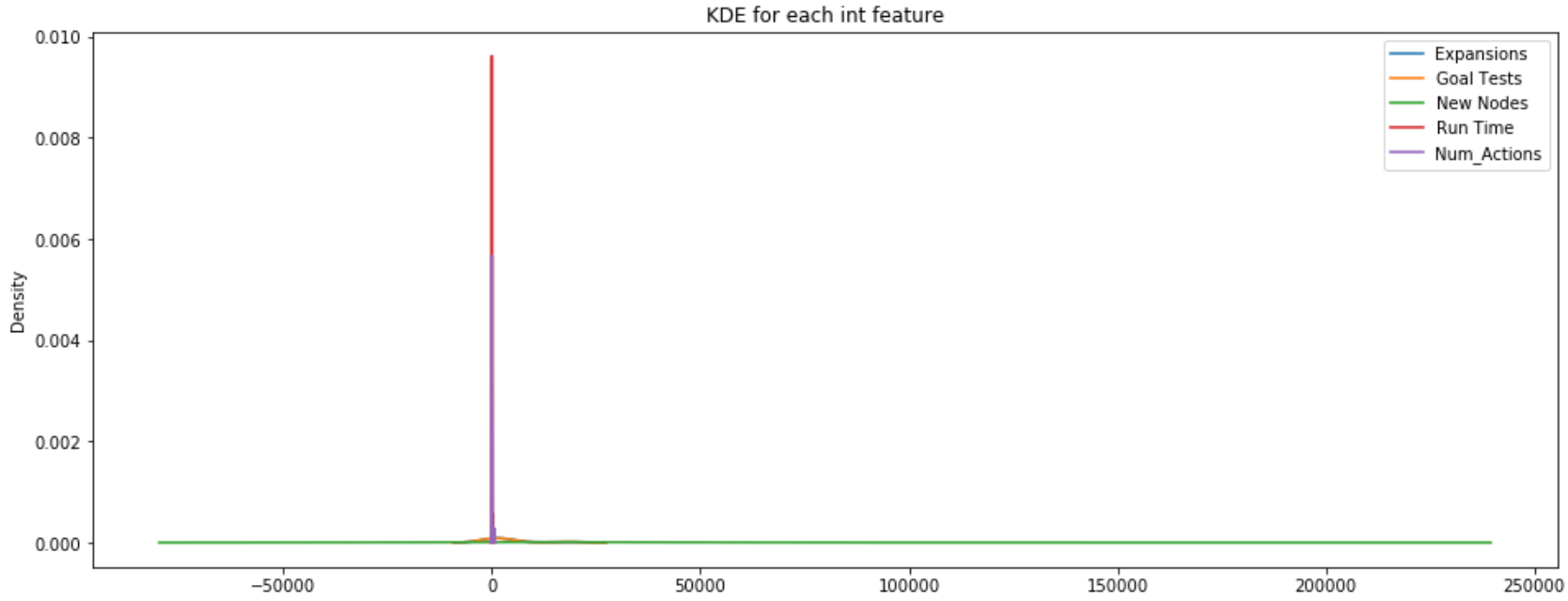
	curr_prob	curr_search	Expansions	Goal Tests	New Nodes	Run Time	Actions	Num_Actions
19	Air Cargo Problem 3	astar_search with h_ignore_preconditions	5040	5042	44944	14.993989	[Load(C2, P2, JFK), Fly(P2, JFK, ORD), Load(C4...	12
20	Air Cargo Problem 3	astar_search with h_pg_levelsum	318	320	2934	267.925145	[Load(C2, P2, JFK), Fly(P2, JFK, ORD), Load(C4...	12

In [2]:
```
data[data['curr_search']=='breadth_first_search'].plot.box(title = 'boxplot for each int feature',figsize = (16,6))
plt.show()
```

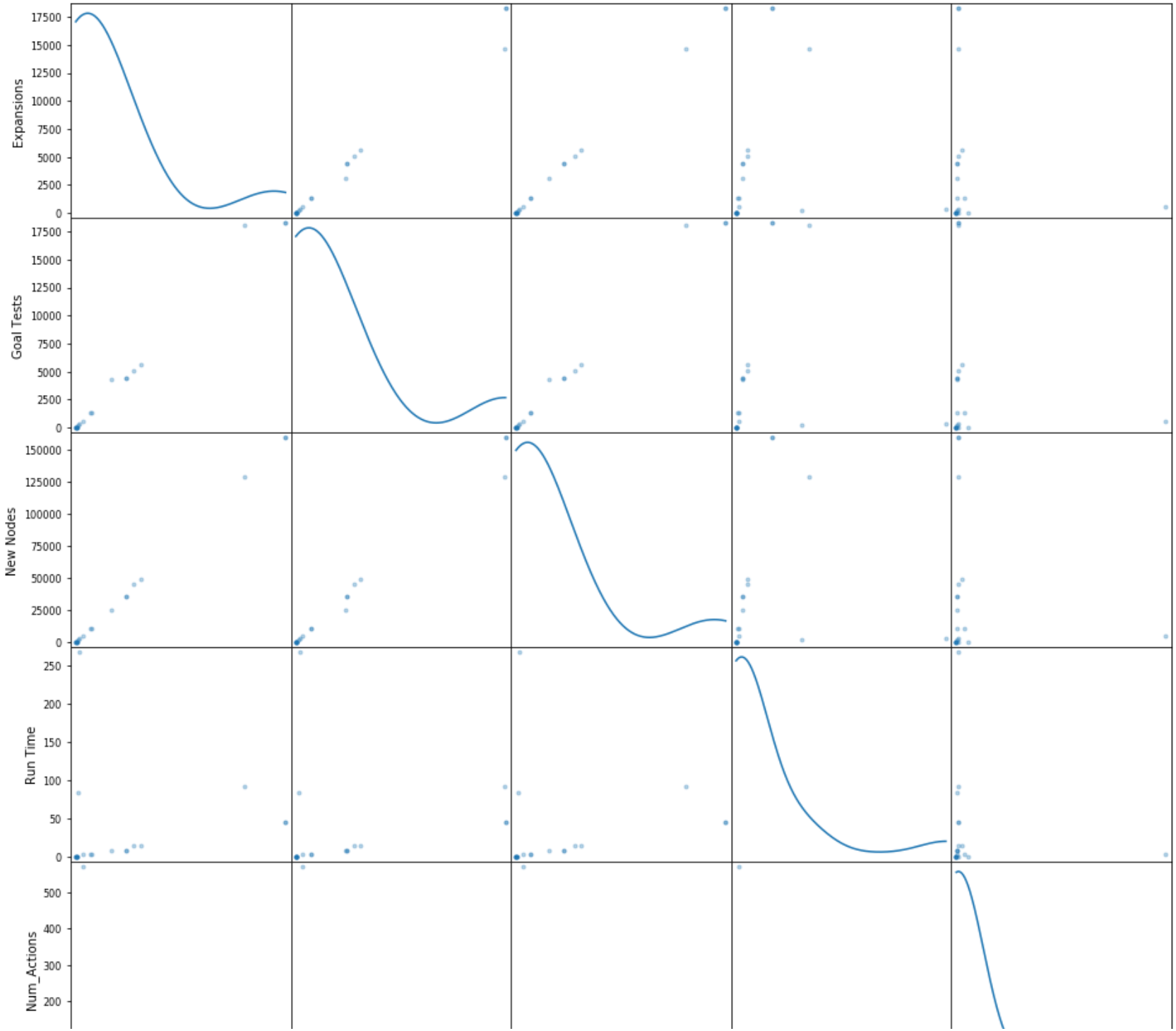


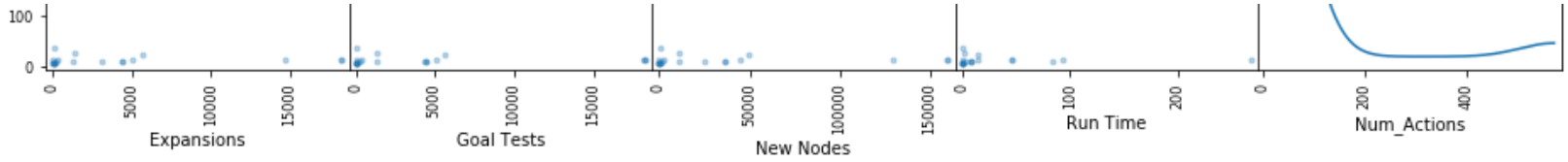
In [3]:
```
data.plot.kde(title = 'KDE for each int feature',figsize = (16,6))
```

Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x173357df780>

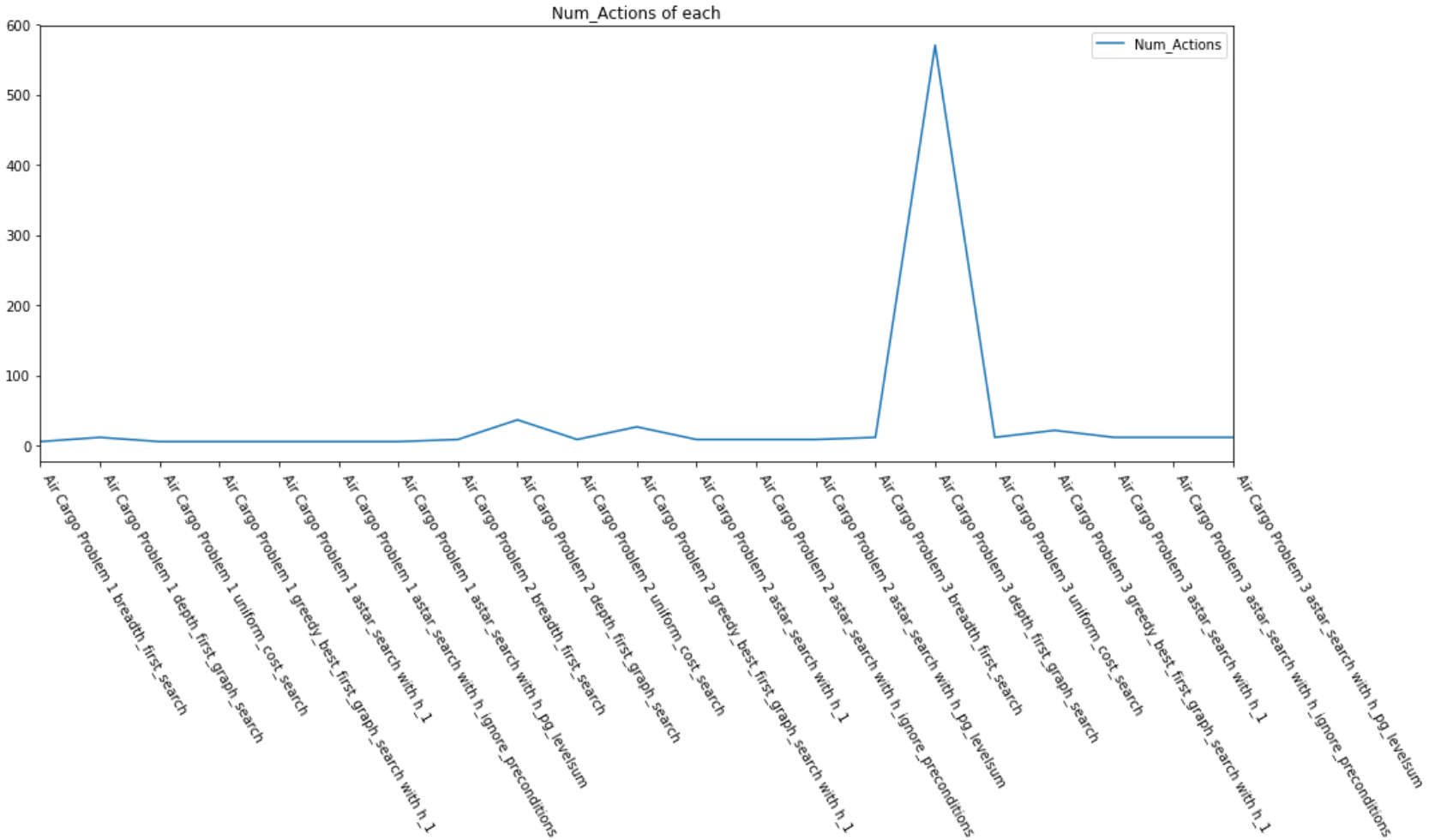


In [4]:
```
pd.scatter_matrix(data, alpha = 0.3, figsize = (16,16), diagonal = 'kde')
plt.show()
```

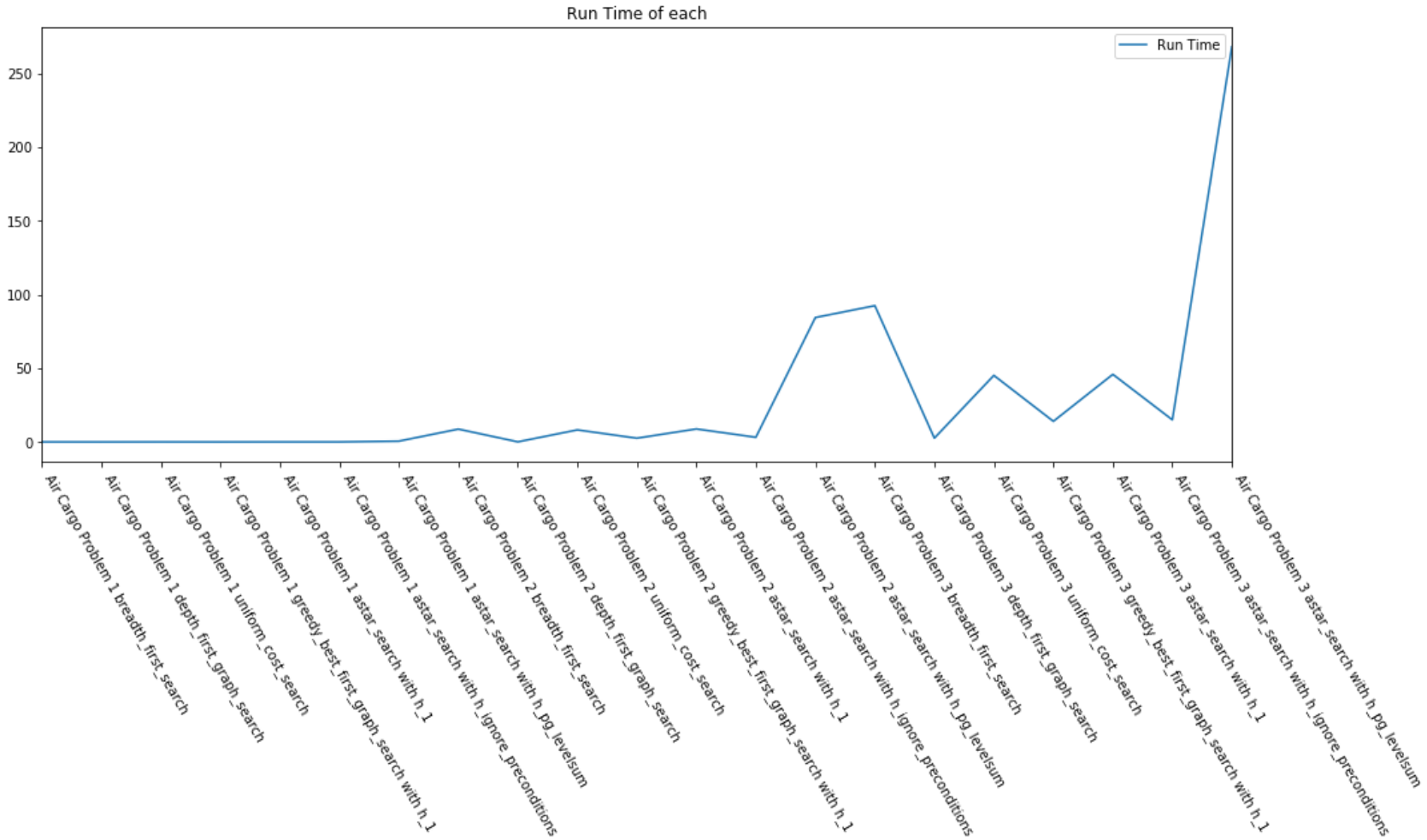

Expansions
Goal Tests
New Nodes
Run Time
Num_Actions
0
2500
5000
7500
10000
12500
15000
17500
0
25000
50000
75000
100000
125000
150000
0
50
100
150
200
250
200
300
400
500



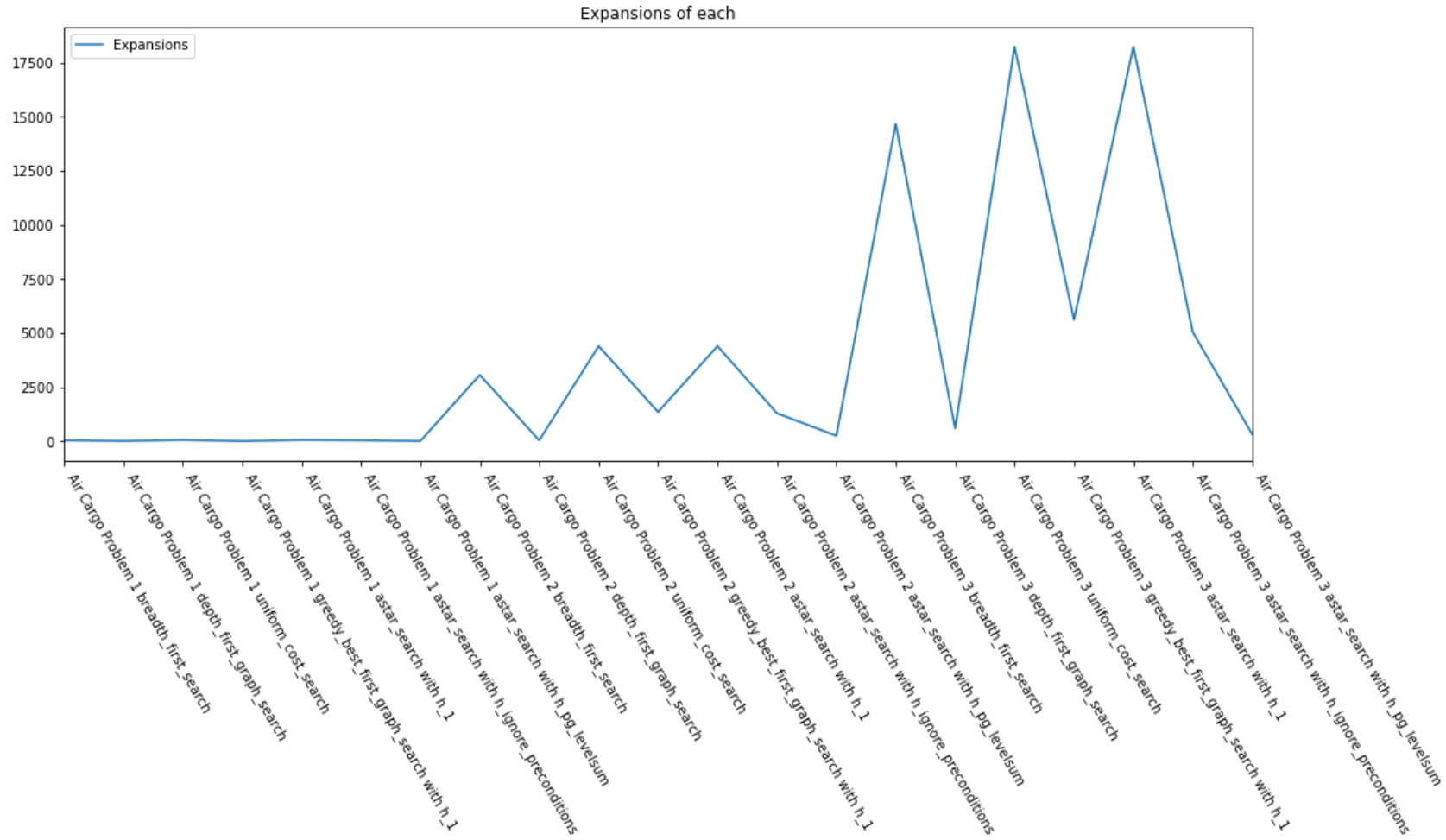
In [5]:
```
ax=data['Num_Actions'].plot(title = 'Num_Actions of each',figsize = (16,6),legend = True)
plt.xticks(range(len(data['Num_Actions'])))
ax.set_xticklabels(data['curr_prob']+" "+data['curr_search'],rotation='300',ha='left')
plt.show()
```



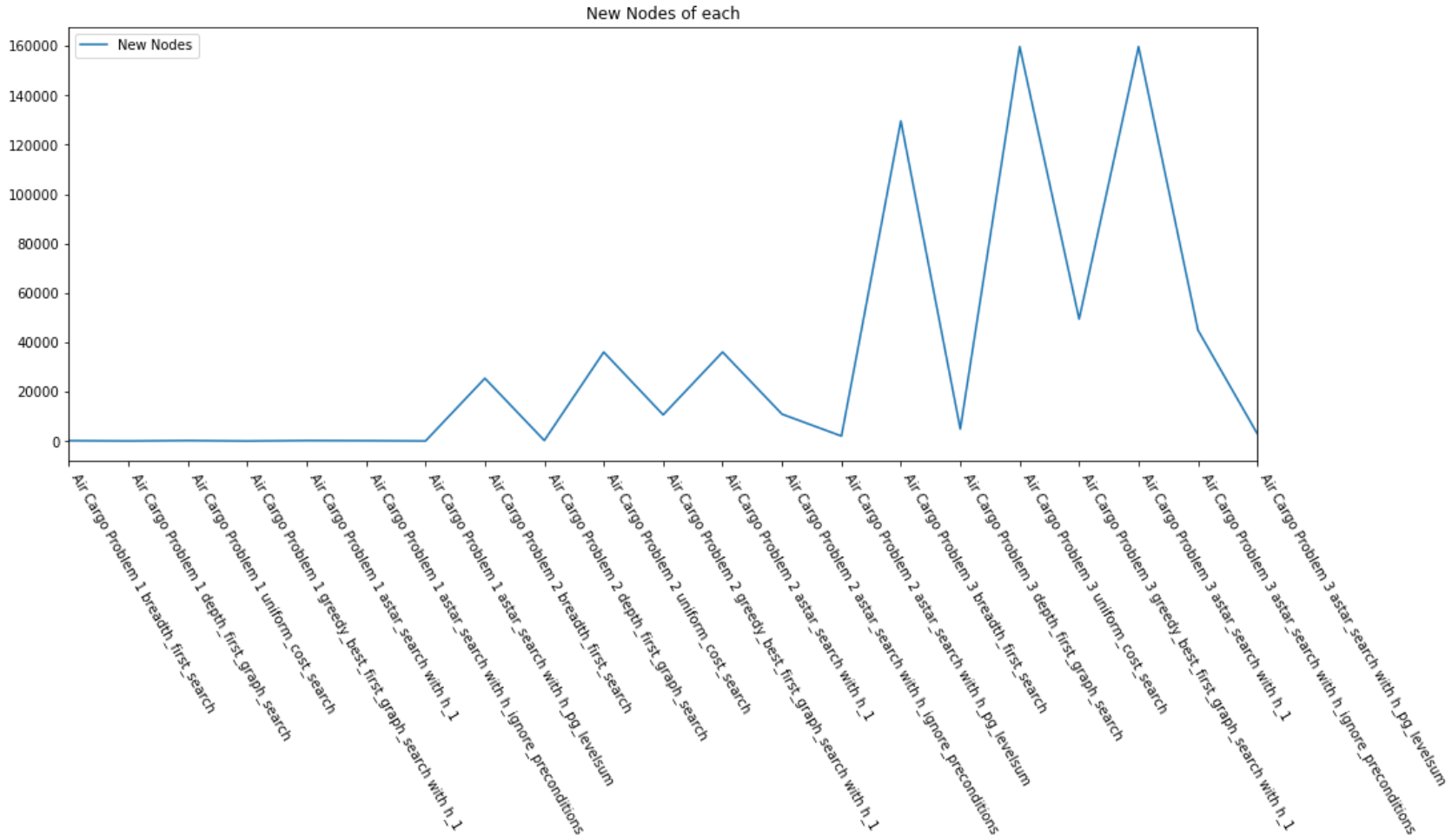
In [6]:
```
ax=data['Run Time'].plot(title = 'Run Time of each',figsize = (16,6),legend = True)
plt.xticks(range(len(data['Run Time'])))
ax.set_xticklabels(data['curr_prob']+" "+data['curr_search'],rotation='300',ha='left')
plt.show()
```



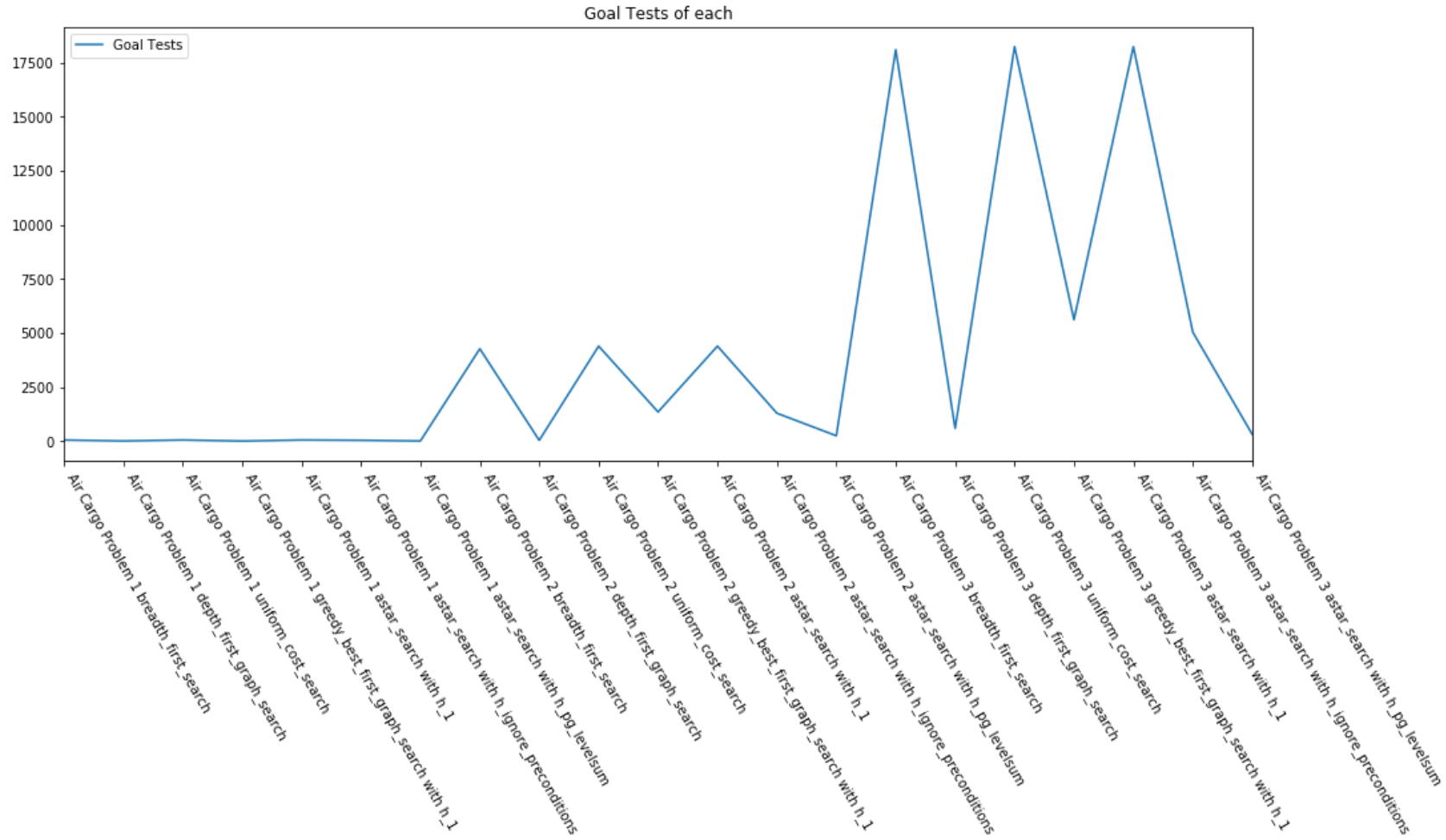
In [7]:
```
ax=data['Expansions'].plot(title = 'Expansions of each',figsize = (16,6),legend = True)
plt.xticks(range(len(data['Expansions'])))
ax.set_xticklabels(data['curr_prob']+" "+data['curr_search'],rotation='300',ha='left')
plt.show()
```



In [8]:
```
ax=data['New Nodes'].plot(title = 'New Nodes of each',figsize = (16,6),legend = True)
plt.xticks(range(len(data['New Nodes'])))
ax.set_xticklabels(data['curr_prob']+" "+data['curr_search'],rotation='300',ha='left')
plt.show()
```



In [9]:
```
ax=data['Goal Tests'].plot(title = 'Goal Tests of each',figsize = (16,6),legend = True)
plt.xticks(range(len(data['Goal Tests'])))
ax.set_xticklabels(data['curr_prob']+" "+data['curr_search'],rotation='300',ha='left')
plt.show()
```



Analysis

Non-Heuristic search

With regards to Breadth_First/Depth_First/Uniform cost seach, we are able to observe:

- Depth First has best run time, but non-optimal plans for all problems
- Breadth First and Uniform Cost both acheive optimal plans
- as problem complexity goes up Breadth First and Uniform Complexity trade place for best run time
    - For low complexity Breadth First performs slightly better, and either search may be chosen
    - for higher complexity Uniform Cost should be chosen, as plan results are optimal but runtime doesn't increase as sharply

Heuristic search

When evaluating the results of greedy_best_first_graph_search and the A* searches, we see:

- greedy_best_first_graph_search, astar_search with h_ignore_preconditions have the best run times.
- greedy_best_first_graph_search does not reach an optimal plan
- all A* results reach an optimal plan
- Due to best run time and optimal plan, astar_search with h_ignore_preconditions is the best option

All search

When we look at the results of all the data, we can see namy things:

- Depth_first, uniform_cost, and all A* reach the optimum plan for the problem
- The run time and results do not seem to be co-related with new nodes/expansion/goal tests. More problem sets are needed for this to be clear
- Best run times are acheived by depth_first_graph_search, greedy_best_first_graph_search, astar_search with h_ignore_preconditions
    - Of the best run times, the A* search is the only one to reach an optimum plan, and should be first choice for most situations.
- the longest plan is acheived by depth_first_graph_search, although it does have the best runtime

EOF