# Import Packages

In [1]:
```
import pyomo.environ as pe
import json
from math import floor, ceil

# plotting
import matplotlib.pyplot as plt
import networkx as nx
from networkx.drawing.nx_pylab import draw_networkx
import plotly
import plotly.graph_objs as go
```

In [2]:
```
model = pe.ConcreteModel(name='815Project')
```

# Import Data

In [3]:
```
data = pe.DataPortal()
folder_name = './processed_data_10_nodes_yes_time_window/'
filename = "lc101.dat"
data.load(filename=folder_name+filename)
```

In [4]:
```
with open((folder_name+filename).replace('.dat','.json'), 'r') as fp:
    location_data = json.load(fp)
```

# Declare Sets

In [5]:
```
# Total nodes
model.N = pe.Set(initialize=data['N'])
# total vehicles
model.K = pe.Set(initialize=data['K'])
# total requests
model.R = pe.Set(initialize=data['R'])
# transshipment node
model.T = pe.Set(within=model.N,initialize=data['T'])
```

# Initialize Parameters

In [6]:
```
# load carrying capacity
model.u = pe.Param(model.K,within=pe.NonNegativeReals,initialize=data['u'])
# initial depot
model.o = pe.Param(model.K,within=model.N,initialize=data['o'])
# final depot
model.o_ = pe.Param(model.K,within=model.N,initialize=data['o_'])
# request quantity
model.q = pe.Param(model.R,within=pe.NonNegativeReals,initialize=data['q'])
# request pick up
model.p = pe.Param(model.R,within=model.N,initialize=data['p'])
# request drop pff
model.d = pe.Param(model.R,within=model.N,initialize=data['d'])
# transport cost
model.c = pe.Param(model.N,model.N,model.K,within=pe.NonNegativeReals,initialize=0,mutable=True)
# transport time
model.tau = pe.Param(model.N,model.N,model.K,within=pe.NonNegativeReals,initialize=0,mutable=True)
```

## time related parameters

In [7]:
```
# earliest time
model.t_e = pe.Param(model.N,initialize=data['t_e'])
# latest time
model.t_l = pe.Param(model.N,initialize=data['t_l'])
```

In [8]:
```
# put values to cost and time
data = pe.DataPortal()
data.load(filename=folder_name+filename,model=model)
for index,value in data['c'].items():
    model.c[index] = value
for index,value in data['tau'].items():
    model.tau[index] = value
# total arcs
def arcs_rule(model,i,j):
    return model.c[i,j,1].value > 0
model.A = pe.Set(initialize=model.N*model.N, filter = arcs_rule)
```

# Declare Variables

In [9]:
```
# if a car K travels in arc A
model.x = pe.Var(model.A,model.K,within=pe.Binary)
# if a car K travels in arc A carries order R
model.y = pe.Var(model.A,model.K,model.R,within=pe.Binary)
# if node i precedes (not necessarily immediately) node j in the route of the vehicle k
model.z = pe.Var(model.A,model.K,within=pe.Binary)
```

### time related variables

In [10]:
```
# actual arrival time
model.t_a = pe.Var(model.N,model.K)
# actual departure time
model.t_d = pe.Var(model.N,model.K)
# if transshipment happens
model.s = pe.Var(model.T,model.R,model.K,model.K,within=pe.Binary)
```

# Write Constraints

(1) enforce that each vehicle may initiate at most one route from its origin depot; constraints

In [11]:
```
def one_car_start_rule(model,i,k):
    if i == model.o[k]:
        return sum(model.x[i,j,k] for j in model.N if (i,j) in model.A) <= 1
    else:
        return pe.Constraint.Skip
model.one_car_start_con = pe.Constraint(model.N,model.K,rule=one_car_start_rule)
```

#### Special Note: Addition (correction) from paper formulation

In [12]:
```
def no_car_enter_rule(model,i,k):
    if i == model.o[k]:
        return sum(model.x[j,i,k] for j in model.N if (j,i) in model.A) == 0
    else:
        return pe.Constraint.Skip
model.no_car_enter_con = pe.Constraint(model.N,model.K,rule=no_car_enter_rule)
```

(2) enforce that the same vehicle must end the route at its final depot.

In [13]:
```
def end_at_final_rule(model,i,l,k):
    if i == model.o[k] and l == model.o_[k]:
        return sum(model.x[i,j,k] for j in model.N if (i,j) in model.A) \
                == sum(model.x[j,l,k] for j in model.N if (j,l) in model.A)
    else:
        return pe.Constraint.Skip
model.end_at_final_con = pe.Constraint(model.N,model.N,model.K,rule=end_at_final_rule)
```

#### Special Note: Addition (correction) from paper formulation

In [14]:
```
def no_car_leave_rule(model,i,k):
    if i == model.o_[k]:
        return sum(model.x[i,j,k] for j in model.N if (i,j) in model.A) == 0
    else:
        return pe.Constraint.Skip
model.no_car_leave_con = pe.Constraint(model.N,model.K,rule=no_car_leave_rule)
```

(3) maintain flow conservation of the vehicles through the nodes in the network.

In [15]:
```
def flow_conservation_rule(model,i,k):
    if i != model.o[k] and i != model.o_[k]:
        return sum(model.x[i,j,k] for j in model.N if (i,j) in model.A) \
                == sum(model.x[j,i,k] for j in model.N if (j,i) in model.A)
    else:
        return pe.Constraint.Skip
model.flow_conservation_con = pe.Constraint(model.N,model.K,rule=flow_conservation_rule)
```

(4) & (5) enforce all pickups and deliveries of the customer requests.

In [16]:
```
def pickup_request_rule(model,i,r):
    if i == model.p[r]:
        return sum(model.y[i,j,k,r] for k in model.K for j in model.N if (i,j) in model.A) == 1
    else:
        return pe.Constraint.Skip
model.pickup_request_con = pe.Constraint(model.N,model.R,rule=pickup_request_rule)
```

In [17]:
```
def deliver_request_rule(model,i,r):
    if i == model.d[r]:
        return sum(model.y[j,i,k,r] for k in model.K for j in model.N if (j,i) in model.A) == 1
    else:
        return pe.Constraint.Skip
model.deliver_request_con = pe.Constraint(model.N,model.R,rule=deliver_request_rule)
```

(6) maintain the request flow conservation at the transshipment nodes allowing requests to switch from one vehicle to another while constraints

**Special Note: Deviation (correction) from paper formulation**

In [18]:
```
def trans_conservation_rule(model,i,r):

    how_many_package_to_be_dropped = 1 if i == model.d[r] else 0
    how_many_package_to_be_picked = 1 if i == model.p[r] else 0

    return sum(model.y[i,j,k,r] for k in model.K for j in model.N if (i,j) in model.A) \
            + how_many_package_to_be_dropped \
            == sum(model.y[j,i,k,r] for k in model.K for j in model.N if (j,i) in model.A) \
            + how_many_package_to_be_picked
model.trans_conservation_con = pe.Constraint(model.T,model.R,rule=trans_conservation_rule)
```

(7) maintain the request flow conservation at the non-transshipment nodes requiring that any vehicle bringing a request must also leave carrying the same request.

In [19]:
```
def request_conservation_rule(model,i,k,r):
    if i not in model.T and i != model.p[r] and i != model.d[r]:
        return sum(model.y[i,j,k,r] for j in model.N if (i,j) in model.A) \
                == sum(model.y[j,i,k,r] for j in model.N if (j,i) in model.A)
    else:
        return pe.Constraint.Skip
model.request_conservation_con = pe.Constraint(model.N,model.K,model.R,rule=request_conservation_rule)
```

(8) enforce a vehicle flow on an arc if there is some request flow in the same vehicle on the same arc.

In [20]:
```
def request_needs_car_rule(model,i,j,k,r):
    return model.y[i,j,k,r] <= model.x[i,j,k]
model.request_needs_car_con = pe.Constraint(model.A,model.K,model.R,rule=request_needs_car_rule)
```

(9) ensure capacity of each vehicle on each arc of the network

In [21]:
```
def capacity_rule(model,i,j,k):
    return sum(model.q[r]*model.y[i,j,k,r] for r in model.R) <= model.u[k] * model.x[i,j,k]
model.capacity_con = pe.Constraint(model.A,model.K,rule=capacity_rule)
```

(12,13,14) subtour elimination

In [22]:
```
def immediate_order_rule(model,i,j,k):
    if i == model.o[k] or j == model.o_[k]:
        return pe.Constraint.Skip
    else:
        return model.x[i,j,k] <= model.z[i,j,k]
model.immediate_order_con = pe.Constraint(model.A,model.K,rule=immediate_order_rule)
```

In [23]:
```
def one_direction_ahead_rule(model,i,j,k):
    if i == model.o[k] or j == model.o_[k]:
        return pe.Constraint.Skip
    else:
        return model.z[i,j,k] + model.z[j,i,k] == 1
model.one_direction_ahead_con = pe.Constraint(model.A,model.K,rule=one_direction_ahead_rule)
```

In [24]:
```
def no_triangle_rule(model,i,j,l,k):
    if (i,j) in model.A and (j,l) in model.A and (l,i) in model.A:
        return model.z[i,j,k] + model.z[j,l,k] + model.z[l,i,k] <= 2
    else:
        return pe.Constraint.Skip
model.no_triangle_con = pe.Constraint(model.N,model.N,model.N,model.K,rule=no_triangle_rule)
```

(15,16) timing calculation

In [25]:
```
def travel_time_rule(model,i,j,k):
    return model.t_d[i,k] + model.tau[i,j,k] - model.t_a[j,k] <= 1000*(1-model.x[i,j,k])
model.travel_time_con = pe.Constraint(model.A,model.K,rule=travel_time_rule)
```

In [26]:
```
def arrive_first_rule(model,i,k):
    return model.t_a[i,k] <= model.t_d[i,k]
model.arrive_first_con = pe.Constraint(model.N,model.K,rule=arrive_first_rule)
```

(17,18) pick up and delivery time window

In [27]:
```
def pick_up_arrival_rule(model,r,k):
    return model.t_e[model.p[r]] <= model.t_a[model.p[r],k]
model.pick_up_arrival_con = pe.Constraint(model.R,model.K,rule=pick_up_arrival_rule)

def pick_up_depart_rule(model,r,k):
    return model.t_l[model.p[r]] >= model.t_d[model.p[r],k]
model.pick_up_depart_con = pe.Constraint(model.R,model.K,rule=pick_up_depart_rule)

def delivery_arrival_rule(model,r,k):
    return model.t_e[model.d[r]] <= model.t_a[model.d[r],k]
model.delivery_arrival_con = pe.Constraint(model.R,model.K,rule=delivery_arrival_rule)

def delivery_depart_rule(model,r,k):
    return model.t_l[model.d[r]] >= model.t_d[model.d[r],k]
model.delivery_depart_con = pe.Constraint(model.R,model.K,rule=delivery_depart_rule)
```

(19,20) transshipment time window

In [28]:
```
def transshipment_detector_rule(model,r,i,k,l):
    if k == l:
        return pe.Constraint.Skip
    else:
        return sum(model.y[j,i,k,r] for j in model.N if (j,i) in model.A) \
               + sum(model.y[i,j,l,r] for j in model.N if (i,j) in model.A) \
               <= model.s[i,r,k,l] + 1
model.transshipment_detector_con = pe.Constraint(model.R,model.T,model.K,model.K,rule=transshipment_detector_
rule)
```

In [29]:
```
def transshipment_time_window_rule(model,r,i,k,l):
    if k == l:
        return pe.Constraint.Skip
    else:
        return model.t_a[i,k] - model.t_d[i,l] <= 1000*(1-model.s[i,r,k,l])
model.transshipment_time_window_con = pe.Constraint(model.R,model.T,model.K,model.K,rule=transshipment_time_w
indow_rule)
```

# Declare Objective

In [30]:
```
model.obj = pe.Objective(expr=sum(model.c[i,j,k]*model.x[i,j,k] for i,j in model.A for k in model.K))
```

In [31]:
```
opt = pe.SolverFactory('gurobi')
```

In [32]:
```
opt.solve(model,options={'mipgap':0,'TimeLimit':1000},tee=True);
```

```
Academic license - for non-commercial use only
Read LP format model from file /Users/naienh/Desktop/815/tmpqxmqllme.pyomo.lp
Reading time = 0.11 seconds
x14751: 53551 rows, 14501 columns, 211696 nonzeros
Changed value of parameter mipgap to 0.0
   Prev: 0.0001  Min: 0.0  Max: 1e+100  Default: 0.0001
Changed value of parameter TimeLimit to 1000.0
   Prev: 1e+100  Min: 0.0  Max: 1e+100  Default: 1e+100
Optimize a model with 53551 rows, 14501 columns and 211696 nonzeros
Variable types: 201 continuous, 14300 integer (14300 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+03]
  Objective range  [2e-01, 1e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+03]
Presolve removed 35098 rows and 6579 columns
Presolve time: 0.22s
Presolved: 18453 rows, 7922 columns, 82030 nonzeros
Variable types: 150 continuous, 7772 integer (7772 binary)

Root relaxation: objective 2.625887e+01, 4128 iterations, 0.33 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0   26.25887    0   75          -   26.25887      -     -    0s
     0     0   26.66500    0  116          -   26.66500      -     -    0s
     0     0   26.69500    0   86          -   26.69500      -     -    1s
     0     0   26.71000    0   84          -   26.71000      -     -    1s
     0     0   26.75000    0   84          -   26.75000      -     -    1s
     0     0   26.75000    0   88          -   26.75000      -     -    1s
     0     0   26.76927    0  118          -   26.76927      -     -    1s
     0     0   26.76927    0  117          -   26.76927      -     -    1s
     0     0   26.79780    0  119          -   26.79780      -     -    1s
     0     0   26.89187    0  124          -   26.89187      -     -    1s
     0     0   26.89187    0  126          -   26.89187      -     -    1s
     0     0   27.09853    0  120          -   27.09853      -     -    1s
     0     0   27.09853    0  114          -   27.09853      -     -    1s
     0     0   27.09853    0  116          -   27.09853      -     -    1s
     0     0   27.11499    0  125          -   27.11499      -     -    1s
     0     0   27.13813    0  127          -   27.13813      -     -    1s
     0     0   27.13813    0  134          -   27.13813      -     -    1s
     0     0   27.16316    0  120          -   27.16316      -     -    2s
     0     0   27.16316    0  113          -   27.16316      -     -    2s
     0     0   27.17048    0  102          -   27.17048      -     -    2s
     0     0   27.17048    0  104          -   27.17048      -     -    2s
     0     0   27.18048    0  110          -   27.18048      -     -    2s
     0     0   27.18048    0  110          -   27.18048      -     -    2s
     0     0   27.18381    0  110          -   27.18381      -     -    2s
     0     0   27.18381    0  106          -   27.18381      -     -    2s
     0     0   27.18798    0  105          -   27.18798      -     -    2s
     0     0   27.18905    0  109          -   27.18905      -     -    2s
     0     0   27.18905    0  109          -   27.18905      -     -    2s
     0     0   27.23327    0  120          -   27.23327      -     -    2s
     0     0   27.23365    0  119          -   27.23365      -     -    2s
     0     0   27.23385    0  119          -   27.23385      -     -    2s
     0     0   27.27715    0   78          -   27.27715      -     -    2s
     0     0   27.27715    0   75          -   27.27715      -     -    2s
     0     0   27.27715    0   80          -   27.27715      -     -    2s
     0     0   27.27715    0   82          -   27.27715      -     -    2s
     0     0   27.27715    0   79          -   27.27715      -     -    2s
     0     0   27.27715    0   80          -   27.27715      -     -    2s
     0     0   27.27715    0   70          -   27.27715      -     -    3s
     0     0   27.27715    0   73          -   27.27715      -     -    3s
     0     0   27.27715    0   74          -   27.27715      -     -    3s
H    0     0                       66.1100000   27.27715  58.7%     -    3s
     0     0   27.27715    0   72   66.11000   27.27715  58.7%     -    3s
     0     2   27.27715    0   72   66.11000   27.27715  58.7%     -    3s
H   32    33                       58.9400000   27.39812  53.5%   229    4s
H   34    35                       56.3500000   27.39812  51.4%   217    4s
H   68    73                       55.7000000   27.39812  50.8%   220    5s
H   69    74                       55.1000000   27.39812  50.3%   221    5s
H  104   105                       54.5900000   27.39812  49.8%   210    5s
H  203   196                       54.1600000   27.39812  49.4%   243    9s
   219   213   35.43964   27  104   54.16000   27.39812  49.4%   233   10s
H  465   409                       53.8000000   27.42457  49.0%   193   14s
   542   465   29.18929    7   91   53.80000   27.44280  49.0%   188   15s
  1157   927   29.60844   13  133   53.80000   27.50064  48.9%   152   21s
  1299  1028   31.81123   31  115   53.80000   27.50064  48.9%   150   25s
  1312  1037   33.98584   34  145   53.80000   27.84232  48.2%   148   30s
  1328  1047   28.07388   10  208   53.80000   28.07388  47.8%   146   35s
  1340  1055   28.13913   16  194   53.80000   28.13913  47.7%   145   41s
  1343  1060   28.14029   15  169   53.80000   28.14029  47.7%  19.3   47s
  1345  1064   28.14267   16  206   53.80000   28.14267  47.7%  19.9   54s
  1355  1072   28.16857   18  146   53.80000   28.16857  47.6%  23.5   55s
  1465  1140   29.36382   27  125   53.80000   28.17500  47.6%  41.6   60s
  1576  1217   30.14019   39  126   53.80000   28.17500  47.6%  64.0   65s
  1816  1369   39.40770   56  216   53.80000   28.17500  47.6%  90.8   70s
```

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| H | 2020 | 1337 | | | 44.6700000 | 28.17551 | 36.9% | 107 | 74s |
| H | 2021 | 1224 | | | 41.6500000 | 28.17551 | 32.4% | 107 | 75s |
| | 2308 | 1354 | 35.78667 | 65 | 27 | 41.65000 | 28.17551 | 32.4% | 119 | 80s |
| | 2482 | 1452 | 38.24500 | 100 | 53 | 41.65000 | 28.17551 | 32.4% | 121 | 85s |
| | 2644 | 1507 | 29.73867 | 27 | 89 | 41.65000 | 28.18861 | 32.3% | 122 | 90s |
| | 2834 | 1601 | 36.59500 | 72 | 96 | 41.65000 | 28.40373 | 31.8% | 123 | 96s |
| | 3041 | 1726 | 34.45250 | 54 | 111 | 41.65000 | 28.40373 | 31.8% | 128 | 100s |
| | 3275 | 1830 | 35.40429 | 52 | 122 | 41.65000 | 28.42940 | 31.7% | 135 | 105s |
| | 3408 | 1882 | 40.99500 | 76 | 89 | 41.65000 | 28.43693 | 31.7% | 137 | 111s |
| | 3761 | 2028 | cutoff | 75 | | 41.65000 | 28.50122 | 31.6% | 141 | 116s |
| | 4014 | 2125 | 34.42589 | 49 | 79 | 41.65000 | 28.54610 | 31.5% | 145 | 120s |
| | 4375 | 2364 | 39.31577 | 84 | 110 | 41.65000 | 28.61310 | 31.3% | 152 | 126s |
| | 4691 | 2617 | 40.12081 | 82 | 104 | 41.65000 | 28.63393 | 31.3% | 154 | 132s |
| | 4903 | 2777 | 38.54000 | 49 | 155 | 41.65000 | 28.66100 | 31.2% | 153 | 138s |
| | 5073 | 2909 | 33.92512 | 38 | 252 | 41.65000 | 28.68706 | 31.1% | 153 | 140s |
| | 5332 | 3124 | 39.34083 | 71 | 254 | 41.65000 | 28.70378 | 31.1% | 154 | 147s |
| | 5558 | 3273 | 30.96491 | 23 | 166 | 41.65000 | 28.73731 | 31.0% | 154 | 150s |
| | 6036 | 3702 | 36.03500 | 44 | 106 | 41.65000 | 28.73731 | 31.0% | 156 | 156s |
| | 6526 | 4112 | 37.36353 | 64 | 206 | 41.65000 | 28.75592 | 31.0% | 159 | 162s |
| | 6842 | 4367 | 37.53045 | 58 | 155 | 41.65000 | 28.81659 | 30.8% | 159 | 170s |
| | 7352 | 4783 | 34.35062 | 64 | 120 | 41.65000 | 28.83000 | 30.8% | 160 | 178s |
| | 7714 | 5077 | 33.57667 | 32 | 139 | 41.65000 | 28.95154 | 30.5% | 159 | 182s |
| | 8125 | 5418 | 31.66868 | 37 | 216 | 41.65000 | 28.98077 | 30.4% | 158 | 186s |
| | 8573 | 5797 | 31.77000 | 32 | 45 | 41.65000 | 28.98077 | 30.4% | 158 | 190s |
| | 8875 | 6034 | 34.83017 | 43 | 214 | 41.65000 | 29.02101 | 30.3% | 158 | 195s |
| | 9533 | 6570 | 32.83000 | 54 | 134 | 41.65000 | 29.11567 | 30.1% | 159 | 204s |
| | 9864 | 6823 | 39.01000 | 58 | 181 | 41.65000 | 29.17000 | 30.0% | 161 | 208s |
| | 10299 | 7173 | 30.69462 | 31 | 191 | 41.65000 | 29.17389 | 30.0% | 161 | 214s |
| | 10580 | 7389 | 40.69588 | 71 | 169 | 41.65000 | 29.20527 | 29.9% | 162 | 219s |
| | 10950 | 7669 | 34.20140 | 55 | 138 | 41.65000 | 29.24559 | 29.8% | 164 | 224s |
| | 11305 | 7946 | 37.62000 | 54 | 103 | 41.65000 | 29.28706 | 29.7% | 166 | 229s |
| | 11841 | 8352 | 35.61631 | 32 | 186 | 41.65000 | 29.32769 | 29.6% | 165 | 235s |
| | 12259 | 8695 | 37.29667 | 63 | 56 | 41.65000 | 29.33998 | 29.6% | 167 | 240s |
| | 12802 | 9133 | 38.67500 | 72 | 71 | 41.65000 | 29.35922 | 29.5% | 167 | 246s |
| | 13406 | 9567 | 38.74615 | 61 | 165 | 41.65000 | 29.40167 | 29.4% | 167 | 252s |
| | 13872 | 9929 | 40.44000 | 70 | 124 | 41.65000 | 29.45316 | 29.3% | 168 | 259s |
| | 14383 | 10316 | 36.39575 | 46 | 208 | 41.65000 | 29.46313 | 29.3% | 169 | 266s |
| | 15016 | 10845 | 41.56000 | 43 | 185 | 41.65000 | 29.47273 | 29.2% | 169 | 275s |
| | 15119 | 10913 | 38.77500 | 45 | 133 | 41.65000 | 29.48515 | 29.2% | 169 | 283s |
| | 15890 | 11510 | 33.69897 | 37 | 107 | 41.65000 | 29.54161 | 29.1% | 169 | 290s |
| | 16523 | 11973 | 38.27770 | 70 | 106 | 41.65000 | 29.55134 | 29.0% | 169 | 298s |
| | 16957 | 12284 | 35.71407 | 31 | 242 | 41.65000 | 29.59501 | 28.9% | 170 | 306s |
| | 17633 | 12873 | 39.22528 | 69 | 50 | 41.65000 | 29.61277 | 28.9% | 170 | 319s |
| | 18065 | 13172 | 31.80455 | 35 | 130 | 41.65000 | 29.64571 | 28.8% | 170 | 327s |
| | 18782 | 13760 | 33.43277 | 44 | 207 | 41.65000 | 29.67059 | 28.8% | 170 | 335s |
| | 19349 | 14198 | 35.41000 | 55 | 52 | 41.65000 | 29.68953 | 28.7% | 170 | 343s |
| | 20041 | 14747 | 35.12290 | 33 | 115 | 41.65000 | 29.71087 | 28.7% | 170 | 351s |
| | 20780 | 15302 | 32.63919 | 50 | 135 | 41.65000 | 29.73389 | 28.6% | 170 | 359s |
| | 21307 | 15722 | 32.71987 | 45 | 91 | 41.65000 | 29.74750 | 28.6% | 170 | 367s |
| | 22109 | 16387 | 30.72644 | 22 | 153 | 41.65000 | 29.77000 | 28.5% | 170 | 375s |
| | 22628 | 16798 | 36.44563 | 35 | 148 | 41.65000 | 29.77708 | 28.5% | 171 | 383s |
| | 23135 | 17174 | 36.85443 | 74 | 165 | 41.65000 | 29.79825 | 28.5% | 171 | 391s |
| | 23719 | 17600 | 32.96453 | 36 | 262 | 41.65000 | 29.81611 | 28.4% | 172 | 399s |
| | 24304 | 18056 | 41.01667 | 79 | 145 | 41.65000 | 29.82667 | 28.4% | 172 | 407s |
| | 24985 | 18596 | 36.26000 | 78 | 141 | 41.65000 | 29.84427 | 28.3% | 173 | 414s |
| | 25650 | 19138 | 38.19167 | 50 | 72 | 41.65000 | 29.84427 | 28.3% | 173 | 496s |
| | 25655 | 19141 | 39.34778 | 87 | 176 | 41.65000 | 29.84427 | 28.3% | 173 | 500s |
| | 25663 | 19147 | 34.82987 | 36 | 170 | 41.65000 | 29.84427 | 28.3% | 173 | 505s |
| | 25668 | 19150 | 32.52567 | 27 | 202 | 41.65000 | 29.84427 | 28.3% | 173 | 510s |
| | 25673 | 19153 | 31.53539 | 31 | 181 | 41.65000 | 29.84427 | 28.3% | 173 | 515s |
| | 25677 | 19156 | 38.64829 | 64 | 204 | 41.65000 | 29.84427 | 28.3% | 173 | 521s |
| | 25681 | 19159 | 34.44000 | 33 | 188 | 41.65000 | 29.84427 | 28.3% | 173 | 525s |
| | 25686 | 19162 | 29.95258 | 24 | 235 | 41.65000 | 29.84427 | 28.3% | 173 | 530s |
| | 25691 | 19165 | 32.74156 | 54 | 164 | 41.65000 | 29.84427 | 28.3% | 173 | 536s |
| | 25695 | 19168 | 34.05625 | 35 | 250 | 41.65000 | 29.84427 | 28.3% | 173 | 541s |
| | 25698 | 19170 | 38.95750 | 71 | 223 | 41.65000 | 29.84427 | 28.3% | 172 | 546s |
| | 25702 | 19173 | 39.46000 | 80 | 237 | 41.65000 | 29.84427 | 28.3% | 172 | 550s |
| | 25707 | 19176 | 39.08188 | 59 | 246 | 41.65000 | 29.84427 | 28.3% | 172 | 555s |
| | 25710 | 19178 | 37.51262 | 96 | 263 | 41.65000 | 29.84427 | 28.3% | 172 | 561s |
| | 25714 | 19181 | 30.72790 | 35 | 273 | 41.65000 | 29.84427 | 28.3% | 172 | 565s |
| | 25717 | 19183 | 31.22010 | 32 | 273 | 41.65000 | 29.84427 | 28.3% | 172 | 571s |
| | 25721 | 19185 | 35.47266 | 64 | 257 | 41.65000 | 29.84427 | 28.3% | 172 | 575s |
| | 25724 | 19187 | 40.21667 | 100 | 247 | 41.65000 | 29.84427 | 28.3% | 172 | 582s |
| | 25726 | 19189 | 35.72492 | 51 | 215 | 41.65000 | 29.84427 | 28.3% | 172 | 585s |
| | 25730 | 19191 | 32.87790 | 67 | 246 | 41.65000 | 29.84427 | 28.3% | 172 | 590s |
| | 25734 | 19194 | 38.86958 | 70 | 238 | 41.65000 | 29.84427 | 28.3% | 172 | 595s |
| | 25738 | 19197 | 36.80570 | 48 | 250 | 41.65000 | 29.84427 | 28.3% | 172 | 601s |
| | 25740 | 19198 | 30.67625 | 22 | 250 | 41.65000 | 29.84427 | 28.3% | 172 | 605s |
| | 25743 | 19200 | 35.69674 | 24 | 250 | 41.65000 | 29.84427 | 28.3% | 172 | 610s |
| | 25747 | 19203 | 37.39537 | 58 | 269 | 41.65000 | 29.84427 | 28.3% | 172 | 616s |
| | 25749 | 19204 | 36.62809 | 91 | 243 | 41.65000 | 29.84427 | 28.3% | 172 | 620s |
| | 25752 | 19206 | 33.19712 | 50 | 272 | 41.65000 | 29.84427 | 28.3% | 172 | 625s |
| | 25755 | 19208 | 39.34778 | 87 | 275 | 41.65000 | 29.84427 | 28.3% | 172 | 631s |
| | 25757 | 19209 | 32.53833 | 44 | 285 | 41.65000 | 29.84427 | 28.3% | 172 | 635s |
| | 25763 | 19213 | 34.82987 | 36 | 294 | 41.65000 | 29.84427 | 28.3% | 172 | 640s |
| | 25766 | 19215 | 35.14663 | 57 | 297 | 41.65000 | 29.84427 | 28.3% | 172 | 645s |
| | 25769 | 19217 | 40.48000 | 61 | 265 | 41.65000 | 29.84427 | 28.3% | 172 | 650s |
| | 25774 | 19221 | 37.47111 | 73 | 304 | 41.65000 | 29.84427 | 28.3% | 172 | 655s |
| | 25777 | 19223 | 38.64829 | 64 | 281 | 41.65000 | 29.84427 | 28.3% | 172 | 660s |

```
 25780 19225   36.20333   65  290   41.65000   29.84427  28.3%   172  666s
 25783 19227   39.84833   66  293   41.65000   29.84427  28.3%   172  670s
 25786 19229   29.95258   24  240   41.65000   29.84427  28.3%   172  676s
 25789 19231   41.00000   63  256   41.65000   29.84427  28.3%   172  680s
 25794 19234   31.22000   50  251   41.65000   29.84427  28.3%   172  685s
 25796 19235   32.80743   37  251   41.65000   29.84427  28.3%   172  691s
 25798 19237   38.95750   71  251   41.65000   29.84427  28.3%   172  695s
 25799 19240   29.84427   30  244   41.65000   29.84427  28.3%   174  705s
 25801 19244   29.84427   31  191   41.65000   29.84427  28.3%   174  715s
 25805 19248   29.84427   32  193   41.65000   29.84427  28.3%   174  722s
 25811 19252   29.84427   33  187   41.65000   29.84427  28.3%   174  728s
 25817 19256   30.45045   33  141   41.65000   29.84427  28.3%   174  733s
 25823 19260   29.84427   34  181   41.65000   29.84427  28.3%   175  735s
 25847 19273   29.84427   35  185   41.65000   29.84427  28.3%   175  740s
 25874 19286   29.84427   37  166   41.65000   29.84427  28.3%   175  745s
 25911 19316   29.84427   40  109   41.65000   29.84427  28.3%   176  751s
 25937 19329   30.02551   41  106   41.65000   29.84427  28.3%   176  755s
 26009 19376   30.66257   44  220   41.65000   29.84427  28.3%   176  760s
 26073 19416   29.84427   46  217   41.65000   29.84427  28.3%   177  766s
 26172 19456   31.35336   49  136   41.65000   29.84427  28.3%   177  771s
 26245 19490   31.32031   52  118   41.65000   29.84427  28.3%   177  775s
 26340 19544   31.80044   57   58   41.65000   29.84427  28.3%   178  782s
H26358 18587                       41.4200000   29.84427  27.9%   178  782s
 26435 18603   33.66426   62  176   41.42000   29.84427  27.9%   178  786s
 26566 18636   34.54250   65  137   41.42000   29.84427  27.9%   178  794s
H26570 17718                       41.4100000   29.84427  27.9%   178  794s
 26574 17730   33.75867   66  190   41.41000   29.84427  27.9%   178  796s
 26689 17779   34.38735   71  162   41.41000   29.84427  27.9%   179  800s
 26865 17867   36.07231   78  142   41.41000   29.84427  27.9%   179  805s
 27044 17971     cutoff   90        41.41000   29.84427  27.9%   179  811s
 27280 18076   31.51851   53  107   41.41000   29.84427  27.9%   179  816s
 27453 18164   39.41763   85  107   41.41000   29.84427  27.9%   180  821s
 27673 18276   30.39083   45   85   41.41000   29.84427  27.9%   181  826s
 27782 18327   35.82151   71  135   41.41000   29.84427  27.9%   181  830s
 28113 18443   40.65750   77  142   41.41000   29.84427  27.9%   182  836s
 28541 18610   35.27955   52  183   41.41000   29.84427  27.9%   182  842s
 28750 18726   31.34634   41   53   41.41000   29.84427  27.9%   182  846s
 29220 18944   31.44624   45   54   41.41000   29.84427  27.9%   182  853s
 29464 19066   36.68167   93  175   41.41000   29.84427  27.9%   182  860s
 29663 19185   37.69389  129  100   41.41000   29.84427  27.9%   182  865s
 30258 19428   36.57547  131  231   41.41000   29.84427  27.9%   182  873s
 30513 19555   31.94308   50   93   41.41000   29.84427  27.9%   182  878s
 30829 19706   38.15257  109  118   41.41000   29.84427  27.9%   182  883s
 31154 19837   39.58500   99   72   41.41000   29.84427  27.9%   182  887s
 31520 20016   29.86175   39  146   41.41000   29.84427  27.9%   182  892s
 31865 20150   30.48050   50  145   41.41000   29.84427  27.9%   182  897s
 32083 20259   30.13229   39   55   41.41000   29.84427  27.9%   183  903s
 32482 20449   35.16266  109  142   41.41000   29.84427  27.9%   183  908s
 32869 20662   32.34714   65  146   41.41000   29.84427  27.9%   183  914s
*33055 19862              105       41.1200000   29.84427  27.4%   182  914s
H33229 19038                       40.7100000   29.84427  26.7%   183  920s
 33550 19204   31.03743   37  157   40.71000   29.84427  26.7%   183  926s
 33954 19402   35.20820   56   38   40.71000   29.84427  26.7%   183  932s
 34354 19568     cutoff   48        40.71000   29.84427  26.7%   184  942s
 34701 19746     cutoff   95        40.71000   29.84427  26.7%   184  949s
 35291 19960   34.31032   49  155   40.71000   29.84427  26.7%   184  956s
 35770 20158   32.45373   50  222   40.71000   29.84427  26.7%   184  963s
 36214 20341   32.78240   40  157   40.71000   29.84427  26.7%   185  971s
 36706 20587   31.22250   40  109   40.71000   29.85248  26.7%   185  978s
 37303 20850   37.49696   70  124   40.71000   29.89802  26.6%   186  986s
 37922 21121   32.71924   47  142   40.71000   29.91736  26.5%   186  995s
 38482 21412   32.14920   41  116   40.71000   29.93721  26.5%   186 1000s

Cutting planes:
  Gomory: 21
  Cover: 19
  Implied bound: 32
  MIR: 33
  Flow cover: 66
  Zero half: 91

Explored 38792 nodes (7421616 simplex iterations) in 1000.06 seconds
Thread count was 12 (of 12 available processors)

Solution count 10: 40.71 41.12 41.41 ... 55.1

Time limit reached
Best objective 4.071000000000e+01, best bound 2.995000000000e+01, gap 26.4309%
WARNING: Loading a SolverResults object with an 'aborted' status, but
    containing a solution
```

In [33]:
```
plotly.offline.init_notebook_mode()
```

In [34]:
```
edge_text = dict(x=[],y=[],hovertext=[])
edge_trace = go.Scatter(
    x=[],
    y=[],
    line=dict(width=1,color='#888'),
    hoverinfo='text',
    hovertext=[],
    mode='lines')

for edge in model.A:
    counter = 0
    constructed_string = ['Arc: {} to {}'.format(*edge)]
    for k in model.K:
        if model.x[edge,k] == 1:
            counter += 1
            constructed_string.append('Vehicle {}:'.format(k))
            r_counter = 0
            for r in model.R:
                if model.y[edge,k,r] == 1:
                    r_counter += 1
                    constructed_string.append('\tCarrying Order: {}, {} to {}'.format(r,model.p[r],model.d[r
]))
            if r_counter == 0:
                constructed_string.append('\tCarrying Nothing')

    node1, node2 = edge
    for _ in range(counter):
        x0, y0 = location_data[node1]
        x1, y1 = location_data[node2]
        scale_cofficient = 0.01*(-1)**_*ceil(_/2)
        edge_trace['x'] += tuple([x0+scale_cofficient, x1+scale_cofficient,None])
        edge_trace['y'] += tuple([y0+scale_cofficient, y1+scale_cofficient,None])
        edge_text['x'] += tuple([(1/3*x0+2/3*x1)])
        edge_text['y'] += tuple([(1/3*y0+2/3*y1)])
        edge_text['hovertext'] += tuple(['<br>'.join(constructed_string)])
```

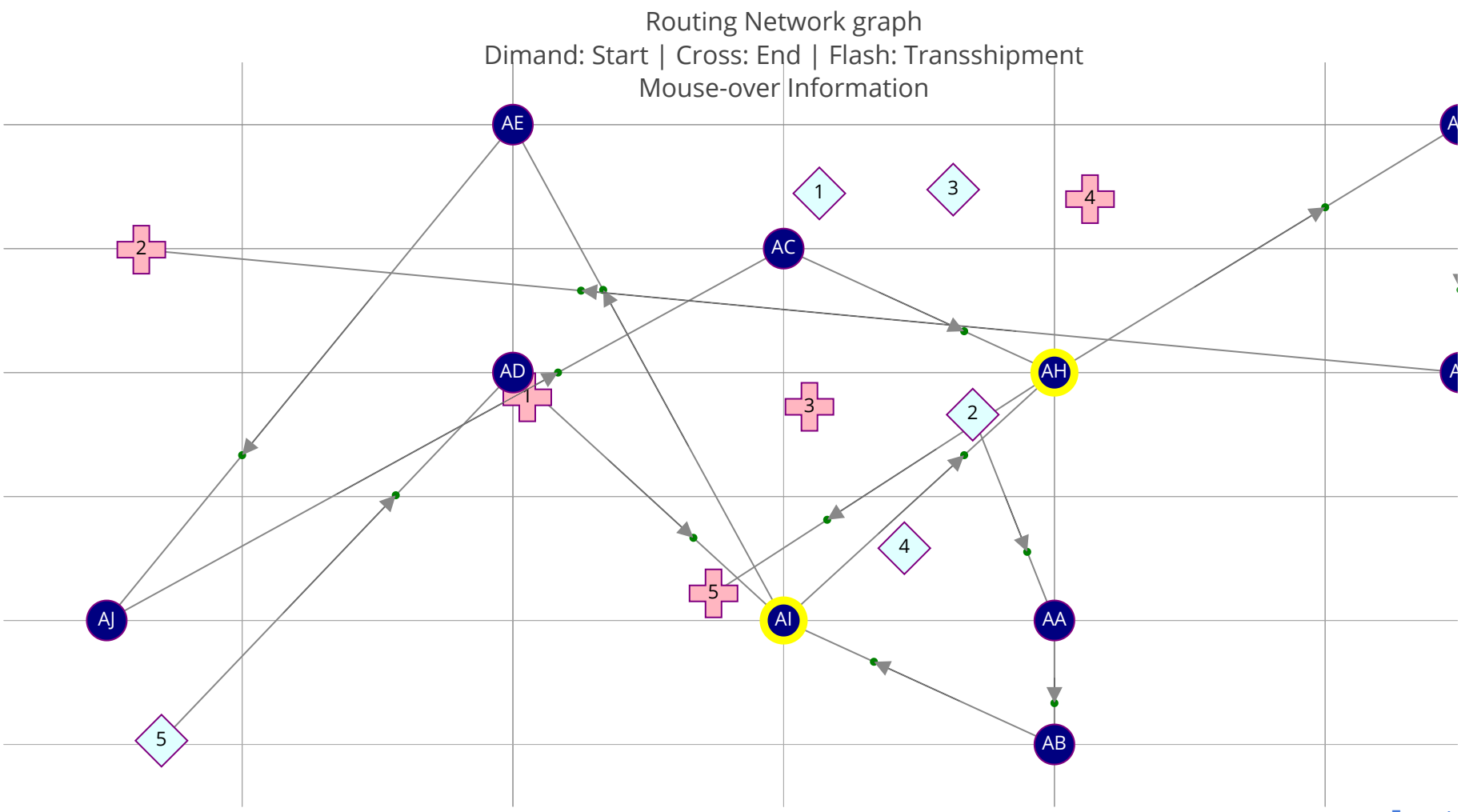
In [35]:
```
center_trace = go.Scatter(
    x=edge_text['x'],
    y=edge_text['y'],
    hoverinfo='text',
    hovertext=edge_text['hovertext'],
    mode='markers',
    marker=dict(
            symbol='circle',
            color='green',
            size=5,
))
```

In [79]:
```
node_trace = go.Scatter(
    x=[],
    y=[],
    text=[],
    textfont=dict(color=[]),
    mode='markers+text',
    hoverinfo='x+y+text',
    hovertext=[],
    marker=dict(
        symbol=[],
        color=[],
        size=25,
        line=dict(width=[],color=[])),
#     selected=dict(marker=dict(color='red',size=30))
)

for node in model.N:
    x, y = location_data[node]
    node_trace['x'] += tuple([x])
    node_trace['y'] += tuple([y])
    # construct doc string
    constructed_string = ['Node: {}'.format(node)]
    counter = 0
    for k in model.K:
        if node == model.o[k]:
            constructed_string.append('Initial Depot: Vehicle {}'.format(k))
            node_trace['marker']['symbol'] += tuple(['diamond'])
            node_trace['marker']['color'] += tuple(['lightcyan'])
            node_trace['marker']['line']['color'] += tuple(['purple'])
            node_trace['marker']['line']['width'] += tuple([1])
            node_trace['text'] += tuple(['{}'.format(k)])
            node_trace['textfont']['color'] += tuple(['black'])
            counter = 1
        if node == model.o_[k]:
            constructed_string.append('End Depot: Vehicle {}'.format(k))
            node_trace['marker']['symbol'] += tuple(['cross'])
            node_trace['marker']['color'] += tuple(['lightpink'])
            node_trace['marker']['line']['color'] += tuple(['purple'])
            node_trace['marker']['line']['width'] += tuple([1])
            node_trace['text'] += tuple(['{}'.format(k)])
            node_trace['textfont']['color'] += tuple(['black'])
            counter = 1
    if counter == 0:
        if node in model.T:
            # check if transshipment happens
            sum19 = [sum(model.y[j,node,k,r].value for j in model.N if (j,node) in model.A)\
                     + sum(model.y[node,j,l,r].value for j in model.N if (node,j) in model.A) == 2\
                     for r in model.R for k in model.K for l in model.K if l != k]
            if sum(sum19) >= 1:
                node_trace['marker']['symbol'] += tuple(['circle'])
                node_trace['marker']['color'] += tuple(['navy'])
                node_trace['marker']['line']['color'] += tuple(['yellow'])
                node_trace['marker']['line']['width'] += tuple([5])
                node_trace['text'] += tuple([node])
                node_trace['textfont']['color'] += tuple(['white'])
                constructed_string.append('Transhipment Node')
            else:
                node_trace['marker']['symbol'] += tuple(['circle'])
                node_trace['marker']['color'] += tuple(['navy'])
                node_trace['marker']['line']['color'] += tuple(['purple'])
                node_trace['marker']['line']['width'] += tuple([1])
                node_trace['text'] += tuple([node])
                node_trace['textfont']['color'] += tuple(['white'])
        else:
            node_trace['marker']['symbol'] += tuple(['circle'])
            node_trace['marker']['color'] += tuple(['navy'])
            node_trace['marker']['line']['color'] += tuple(['purple'])
            node_trace['marker']['line']['width'] += tuple([1])
            node_trace['text'] += tuple([node])
            node_trace['textfont']['color'] += tuple(['white'])
    for r in model.R:
        if node == model.p[r]:
            constructed_string.append('Pick Up: Order {}'.format(r))
        if node == model.d[r]:
            constructed_string.append('Delivery: Order {}'.format(r))

    node_trace['hovertext'] += tuple(['<br>'.join(constructed_string)])
```

In [80]:
```
arrow_dic = [dict(
            x=2/3*edge_trace['x'][3*i+1]+1/3*edge_trace['x'][3*i],
            y=2/3*edge_trace['y'][3*i+1]+1/3*edge_trace['y'][3*i],
            ax=1/3*edge_trace['x'][3*i+1]+2/3*edge_trace['x'][3*i],
            ay=1/3*edge_trace['y'][3*i+1]+2/3*edge_trace['y'][3*i],
            xref='x',
            yref='y',
            axref='x',
            ayref='y',
            showarrow=True,
            arrowhead=2,
            arrowsize=4,
            arrowwidth=0.5,
            arrowcolor='#888',
        ) for i in range(int(len(edge_trace['x'])/3))]
```

In [81]:
```
fig = go.Figure(data=[edge_trace,node_trace,center_trace],
             layout=go.Layout(
                title='Routing Network graph<br>Dimand: Start\t|\tCross: End\t|\tFlash: Transshipment<br>Mouse-over Information',
                titlefont=dict(size=16),
                showlegend=False,
                hovermode='closest',
                margin=dict(b=20,l=5,r=5,t=40),
                xaxis=dict(showgrid=True, zeroline=False, showticklabels=False, hoverformat='.2f'),
                yaxis=dict(showgrid=True, zeroline=False, showticklabels=False, hoverformat='.2f'),
                annotations = arrow_dic))
```

In [82]:
```
plotly.offline.iplot(fig)
```




Export