# Model 1 - Pyomo Exact Solver Code

from pyomo.environ import \*  
  
# Define model  
model = ConcreteModel()  
  
# Links and parameters (from Table 7)  
links = [  
 (1, 12), (1, 5), (12, 6), (12, 8), (4, 5), (5, 6), (6, 7),  
 (7, 8), (4, 9), (4, 9), (5, 9), (6, 10), (7, 11), (8, 2),  
 (9, 10), (10, 11), (11, 2), (11, 3), (13, 3)  
]  
  
free\_flow\_time = {  
 (1, 12): 12, (1, 5): 10, (12, 6): 10, (12, 8): 15, (4, 5): 8,  
 (5, 6): 12, (6, 7): 6, (7, 8): 8, (4, 9): 12, (5, 9): 12,  
 (6, 10): 11, (7, 11): 12, (8, 2): 13, (9, 10): 10,  
 (10, 11): 12, (11, 2): 9, (11, 3): 10, (13, 3): 12  
}  
  
capacity = {  
 (1, 12): 800, (1, 5): 700, (12, 6): 600, (12, 8): 900, (4, 5): 700,  
 (5, 6): 500, (6, 7): 300, (7, 8): 400, (4, 9): 600, (5, 9): 600,  
 (6, 10): 700, (7, 11): 800, (8, 2): 800, (9, 10): 400,  
 (10, 11): 600, (11, 2): 600, (11, 3): 600, (13, 3): 600  
}  
  
# Demand and vulnerability  
total\_demand = 20000  
vulnerable\_demand = 8000  
vul\_ratio = vulnerable\_demand / total\_demand  
EEI\_star = 0.65  
WVPPR\_star = 0.95  
  
model.LINKS = Set(initialize=links)  
model.f = Var(model.LINKS, domain=NonNegativeReals)  
model.z1 = Var(domain=NonNegativeReals)  
model.EEI = Var(domain=NonNegativeReals)  
model.WVPPR = Var(domain=NonNegativeReals)  
  
# Total travel time expression  
def total\_time\_rule(m):  
 return sum(m.f[i]\*free\_flow\_time[i]\*(1 + 0.15\*(m.f[i]/capacity[i])\*\*4) for i in m.LINKS)  
model.total\_time = Expression(rule=total\_time\_rule)  
  
# Vulnerable group flow  
def vulnerable\_flow\_rule(m):  
 return sum(m.f[i]\*vul\_ratio for i in m.LINKS)  
model.vul\_flow = Expression(rule=vulnerable\_flow\_rule)  
  
# EEI and WVPPR constraints  
model.EEI\_def = Constraint(expr=model.EEI \* model.total\_time == total\_demand)  
model.WVPPR\_def = Constraint(expr=model.WVPPR \* vulnerable\_demand == model.vul\_flow)  
  
# Z1 definition  
model.c1 = Constraint(expr=model.z1 <= model.EEI / EEI\_star)  
model.c2 = Constraint(expr=model.z1 <= model.WVPPR / WVPPR\_star)  
  
# Flow conservation (simplified)  
model.flow\_sum = Constraint(expr=sum(model.f[i] for i in model.LINKS) == total\_demand)  
  
# Objective  
model.obj = Objective(expr=model.z1, sense=maximize)