# Model 2 - Pyomo Exact Solver Code

from pyomo.environ import \*  
  
model = ConcreteModel()  
  
# Network  
links = [  
 (1, 12), (1, 5), (12, 6), (12, 8), (4, 5), (5, 6), (6, 7),  
 (7, 8), (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  
}  
  
total\_demand = 20000  
vulnerable\_demand = 8000  
w\_nfii = 0.2  
w\_epe = 0.8  
  
model.LINKS = Set(initialize=links)  
model.f = Var(model.LINKS, domain=NonNegativeReals)  
model.z2 = Var(domain=NonNegativeReals)  
model.EPE = Var(domain=NonNegativeReals)  
model.NFII = Var(domain=NonNegativeReals)  
  
# Evacuation priority metric EPE (assume direct proportionality to flow)  
model.epe\_def = Constraint(expr=model.EPE == sum(model.f[i] for i in model.LINKS) / total\_demand)  
  
# NFII: Need-based fairness inverse index (assumed variation around vulnerable)  
model.nfii\_def = Constraint(expr=model.NFII == sum(abs(model.f[i] - vulnerable\_demand/len(model.LINKS)) for i in model.LINKS) / vulnerable\_demand)  
  
# Z2 = min(EPE / NFII)  
model.z2\_bound = Constraint(expr=model.z2 <= model.EPE / model.NFII)  
  
# Total demand constraint (simplified)  
model.total\_flow = Constraint(expr=sum(model.f[i] for i in model.LINKS) == total\_demand)  
  
model.obj = Objective(expr=model.z2, sense=maximize)