In [14]:

```
from gamspy import (Container, Variable, Equation, Model, Set, Parameter, Su
import numpy as np
```

In [15]:

```
b = Container()

#SETS
districts = Set(b, 'districts', records=['1', '2', '3'])
schools = Set(b, 'schools', records=['East', "West"])

distances = Parameter(b, "distances", domain=[schools, districts], records=r
students = Parameter(b, 'students', domain=districts, records=np.array([250,
minority = Parameter(b, 'minority',domain=districts, records=np.array([50, 5

#VARIABLES
pct_students = Variable(b, "pct_students", "positive", domain=[schools, dist

minority_pct_school = Variable(b, 'minority_pct_school', "positive", domain=


obj = Sum([schools, districts], distances[schools, districts] * pct_students

# City-wide minority percentage
total_students = Sum(districts, students[districts])
total_minority = Sum(districts, minority[districts])
citywide_minority_pct = total_minority / total_students

# Upper and lower bounds for minority percentage per school
upper_minority_pct = citywide_minority_pct + 0.05
lower_minority_pct = citywide_minority_pct - 0.05

minority_balance_upper = Equation(b, 'minority_balance_upper', domain=school
minority_balance_upper[schools] = minority_pct_school[schools] <= upper_mino

minority_balance_lower = Equation(b, 'minority_balance_lower', domain=school
minority_balance_lower[schools] = minority_pct_school[schools] >= lower_mino

# Constraint: Each school must have between 300 and 500 students
enrollment = Sum(districts, pct_students[schools, districts] * students[dist

# Apply the constraints for school enrollment
enrollment_min = Equation(b, 'enrollment_min', domain=[schools])
enrollment_min[:] = enrollment[schools] >= 300

enrollment_max = Equation(b, 'enrollment_max', domain=[schools])
enrollment_max[:] = enrollment[schools] <= 500

total_pct_students = Sum(schools, pct_students[schools, districts])
pct_constraint = Equation(b, 'pct_constraint', domain=districts)
pct_constraint[districts] = total_pct_students == 1




school_model = Model(b,
    name='school_model',
    equations=b.getEquations(),
    problem=Problem.LP,
    sense=Sense.MIN,
    objective=obj)
```

In [16]:

```
school_model.solve(options=Options(equation_listing_limit=100))
print("Objective Function Value: ",round(school_model.objective_value,4),"\r
print("student distribution: \n", pct_students.records)
print("status: ", school_model.status)
print("solver status: ", school_model.solve_status)
```

```
Objective Function Value:  875.0

student distribution:
   schools districts  level  marginal  lower  upper  scale
0     East         1    1.0       0.0    0.0    inf    1.0
1     East         2    0.0     120.0    0.0    inf    1.0
2     East         3    0.2       0.0    0.0    inf    1.0
3     West         1    0.0     325.0    0.0    inf    1.0
4     West         2    1.0       0.0    0.0    inf    1.0
5     West         3    0.8       0.0    0.0    inf    1.0
status:  ModelStatus.OptimalGlobal
solver status:  SolveStatus.NormalCompletion
```

In [ ]: