In [1]:

```
# sets of scools (S) and areas (A)
S = 1:3
A = 1:5

# number of students in each district
n = Dict(zip(A,[1200 1000 1700 2000 2500]))

# capacity of each school
b = Dict(zip(S,[3900 3100 2100]))

using NamedArrays
# distances between district/school pairs
d_matrix = [2.7 0.5 1.6;
    1.4 0.7 2.0;
    2.4 2.9 0.1;
    1.1 0.8 1.3;
    0.5 1.9 2.2]
d_NA = NamedArray(d_matrix,(A,S),("district","school"))

# fraction minority students in the district
g = Dict(zip(A,[0.2 0.1 0.85 0.6 0.9]))

# minimum minority fraction in each school
L = 0.25
# maximum minority fraction in each school
U = 0.8

# minimum enrollment in each school
K = 200;
```

In [2]:

```
using JuMP, Cbc

m = Model()


@variable(m, x[A,S] >= 0) # total students sent from area to school
@variable(m, y[S] >= 0) # total students sent to each school

@objective(m, Min, sum(x[a,s]*d_NA[a,s] for a in A, s in S))

@constraint(m, def_y[s in S], sum(x[a,s] for a in A) == y[s])
@constraint(m, assign_all[a in A], sum(x[a,s] for s in S) == n[a])
@constraint(m, school_cap[s in S], y[s] <= b[s])
@constraint(m, upper_cap[s in S], sum(g[a]*x[a,s] for a in A) <= U*y[s])
@constraint(m, lower_cap[s in S], sum(g[a]*x[a,s] for a in A) >= L*y[s])
@constraint(m, min_enroll[s in S], y[s] >= K)
set_optimizer(m, Cbc.Optimizer)

optimize!(m)
```

```
Presolve 14 (-6) rows, 18 (0) columns and 69 (-6) elements
0  Obj 0 Primal inf 12685.359 (11)
13  Obj 4810
Optimal - objective value 4810
After Postsolve, objective 4810, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 4810 - 13 iterations time 0.002, Presolve 0.00
```

In [ ]: