A communication company plans to place 10 broadcast stations at given locations $a^1, \ldots, a^{10}$ in 10 regions. Each station has its own maximum *signal converge area* of radius r_i (in feet). The radius r_i of each station *i* is at least $\underline{r}$ feet and at most $\overline{r}$ feet (*i* = 1,..., 10). To cover each square foot of the signal converage area of the *i*-th station, a cost of c_i dollars is incurred. Tge company can remotely control these stations by placing a central station at location *x*. This location should be covered by all stations a^i in their *signal coverage area*, but it must be at most r_0 feet far from the company's headquarters located at a given place a_0. The goal is to find the location *x* of the central station and the radius r_i of the *i*-th station to minimize the total cost.

The input data is given as follows:

- The location of all stations is a^1 = (0,4), a^2 = (1,5), a^3 = (2,3), a^4 = (2,1), a^5 = (3,6), a^6 = (4,5), a^7 = (4,1), a^8 = (5,2), a^9 = (6,5), a^{10} = (7,4).
- The location of the headquarters a^0 = (4,4), the upper and lower bounds of the radius $r_0 = 1$, $\overline{r} = 5$, and $\underline{r} = 0.02$.
- The unit costs are *c* = $[1; 2; 1.2; 2.5; 2.1; 1.1; 1.8; 1.4; 1.35; 1.82]^T$

The units of the distances $r_i, \overline{r}, \underline{r}, r_0$ and the coordinates of a^i are in 10^5 feet and the unit cost c_i are in dollars per square foot. The distance is measured by the Euclidean distance, which is defined as $\|x - y\| = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ for any two points $x = (x_1, x_2)$ and $y = (y_1, y_2)$ in R^2.

In [ ]:
```
using JuMP, Ipopt, LinearAlgebra

m = Model(Ipopt.Optimizer)
```

Out[ ]:
```
A JuMP Model
Feasibility problem with:
Variables: 0
Model mode: AUTOMATIC
CachingOptimizer state: EMPTY_OPTIMIZER
Solver name: Ipopt
```

In [ ]:

```
# defining variables
costs = [1; 2; 1.2; 2.5; 2.1; 1.1; 1.8; 1.4; 1.35; 1.82]

a0 = [4, 4]

a1 = [0, 4]
a2 = [1, 5]
a3 = [2, 3]
a4 = [2, 1]
a5 = [3, 6]
a6 = [4, 5]
a7 = [4, 1]
a8 = [5, 2]
a9 = [6, 5]
a10 = [7, 4]

r0 = 1

r_upper = 5
r_lower = 0.02

# variable to represent the radius of each station
stations = 1:10
@variable(m, r_upper >= radius[stations] >= r_lower)

# variable to represent the location of the central station
@variable(m, x[1:2])
```

Out[ ]:

```
2-element Vector{VariableRef}:
 x[1]
 x[2]
```

In [ ]:

```
# defining constraints

@NLexpression(m, expr1, sqrt((x[1]-a1[1])^2 + (x[2]-a1[2])^2))
@NLexpression(m, expr2, sqrt((x[1]-a2[1])^2 + (x[2]-a2[2])^2))
@NLexpression(m, expr3, sqrt((x[1]-a3[1])^2 + (x[2]-a3[2])^2))
@NLexpression(m, expr4, sqrt((x[1]-a4[1])^2 + (x[2]-a4[2])^2))
@NLexpression(m, expr5, sqrt((x[1]-a5[1])^2 + (x[2]-a5[2])^2))
@NLexpression(m, expr6, sqrt((x[1]-a6[1])^2 + (x[2]-a6[2])^2))
@NLexpression(m, expr7, sqrt((x[1]-a7[1])^2 + (x[2]-a7[2])^2))
@NLexpression(m, expr8, sqrt((x[1]-a8[1])^2 + (x[2]-a8[2])^2))
@NLexpression(m, expr9, sqrt((x[1]-a9[1])^2 + (x[2]-a9[2])^2))
@NLexpression(m, expr10, sqrt((x[1]-a10[1])^2 + (x[2]-a10[2])^2))

@NLconstraint(m, distance1, expr1 <= radius[1])
@NLconstraint(m, distance2, expr2 <= radius[2])
@NLconstraint(m, distance3, expr3 <= radius[3])
@NLconstraint(m, distance4, expr4 <= radius[4])
@NLconstraint(m, distance5, expr5 <= radius[5])
@NLconstraint(m, distance6, expr6 <= radius[6])
@NLconstraint(m, distance7, expr7 <= radius[7])
@NLconstraint(m, distance8, expr8 <= radius[8])
@NLconstraint(m, distance9, expr9 <= radius[9])
@NLconstraint(m, distance10, expr10 <= radius[10])

# location of center station must be within r_0 of headquarters
@NLexpression(m, expr0, sqrt((x[1]-a0[1])^2 + (x[2]-a0[2])^2))
@NLconstraint(m, hqdistance, expr0 <= r0)
```

Out[ ]:

$$subexpression_{11} - 1.0 \leq 0$$

In [ ]:

```
# objective function
cost1 = (10^10 * costs[1]) * (pi * radius[1]^2)
cost2 = (10^10 * costs[2]) * (pi * radius[2]^2)
cost3 = (10^10 * costs[3]) * (pi * radius[3]^2)
cost4 = (10^10 * costs[4]) * (pi * radius[4]^2)
cost5 = (10^10 * costs[5]) * (pi * radius[5]^2)
cost6 = (10^10 * costs[6]) * (pi * radius[6]^2)
cost7 = (10^10 * costs[7]) * (pi * radius[7]^2)
cost8 = (10^10 * costs[8]) * (pi * radius[8]^2)
cost9 = (10^10 * costs[9]) * (pi * radius[9]^2)
cost10 = (10^10 * costs[10]) * (pi * radius[10]^2)

@objective(m, Min, cost1+cost2+cost3+cost4+cost5+cost6+cost7+cost8+cost9+cost
```

Out[ ]:

$$3.141592653589793e10 radius_1^2 + 6.283185307179586e10 radius_2^2 + 3.7699111843077$$

$$+ 6.597344572538566e10 radius_5^2 + 3.4557519189487724e10 radius_6^2 + 5.6548667764$$

$$+ 4.2411500823462204e10 radius_9^2 + 5.71769862953$$

In [ ]:

```
# objective goal is to minimize the total cost
optimize!(m)
```

```
This is Ipopt version 3.14.4, running with linear solver MUMPS 5.4.1.

Number of nonzeros in equality constraint Jacobian...:        0
Number of nonzeros in inequality constraint Jacobian.:       32
Number of nonzeros in Lagrangian Hessian.............:       43

Total number of variables............................:       12
                     variables with only lower bounds:        0
                variables with lower and upper bounds:       10
                     variables with only upper bounds:        0
Total number of equality constraints.................:        0
Total number of inequality constraints...............:       11
        inequality constraints with only lower bounds:        0
   inequality constraints with lower and upper bounds:        0
        inequality constraints with only upper bounds:       11

iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr
ls
   0  4.6002311e+08 8.03e+00 1.10e+02  -1.0 0.00e+00    -  0.00e+00 0.00e+00
0
   1  4.5732994e+08 8.01e+00 1.09e+02  -1.0 4.31e+00    -  3.29e-03 3.98e-03h
1
   2  4.2395454e+08 7.32e+00 1.01e+02  -1.0 6.74e+00    -  3.88e-03 8.62e-02f
1
   3  4.1411690e+08 7.14e+00 9.85e+01  -1.0 6.06e+00    -  1.96e-02 2.39e-02f
1
   4  3.8507458e+08 6.53e+00 9.15e+01  -1.0 5.50e+00    -  5.86e-03 8.66e-02f
1
   5  3.8287968e+08 6.39e+00 8.75e+01  -1.0 5.15e+00    -  1.00e-01 2.04e-02f
1
   6  4.0934548e+08 4.78e+00 8.12e+01  -1.0 5.38e+00    -  7.82e-02 2.60e-01h
1
   7  4.7800598e+08 4.32e+00 6.33e+01  -1.0 1.17e+01    -  1.38e-01 1.11e-01h
1
   8  5.2854187e+08 4.28e+00 5.49e+01  -1.0 1.73e+01    -  7.70e-02 1.00e-02h
1
   9  5.3688306e+08 4.28e+00 1.47e+02  -1.0 7.10e+00    -  1.22e-02 2.87e-04h
1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr
ls
  10  2.3308338e+09 4.17e+00 9.25e+01  -1.0 5.21e+00    -  2.32e-03 2.63e-02h
1
  11  4.8299327e+09 4.08e+00 4.75e+02  -1.0 5.04e+00    -  7.86e-02 2.12e-02h
1
  12  2.4409258e+10 3.73e+00 2.43e+02  -1.0 4.93e+00    -  6.53e-02 8.64e-02h
1
  13  2.4689500e+10 3.72e+00 8.72e+02  -1.0 4.54e+00    -  7.64e-02 9.61e-04h
1
  14  2.6669801e+10 3.70e+00 8.32e+02  -1.0 5.19e+00    -  1.70e-03 6.51e-03h
1
  15  2.6705255e+10 3.70e+00 8.47e+02  -1.0 4.13e+00    -  3.70e-03 1.13e-04h
1
  16  3.6870057e+10 3.60e+00 7.25e+02  -1.0 4.21e+00    -  4.74e-04 2.70e-02h
1
  17  3.6994909e+10 3.60e+00 8.35e+02  -1.0 3.84e+00    -  2.74e-02 2.97e-04h
1
  18  5.5313596e+10 3.47e+00 6.24e+02  -1.0 3.35e+00    -  6.59e-04 3.55e-02h
```

```
1
  19  9.0299470e+10 3.30e+00 1.92e+03  -1.0 3.23e+00    -  2.03e-01 5.02e-02h
1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr
ls
  20  9.0729436e+10 3.29e+00 2.33e+03  -1.0 2.92e+00    -  4.89e-02 5.56e-04h
1
  21  1.0309463e+11 3.24e+00 2.18e+03  -1.0 2.93e+00    -  5.77e-04 1.51e-02h
1
  22  1.3595779e+11 3.13e+00 3.43e+03  -1.0 2.82e+00    -  1.78e-01 3.53e-02h
1
  23  1.3704573e+11 3.13e+00 3.44e+03  -1.0 3.19e+00    -  3.01e-03 1.10e-03h
1
  24  4.5014955e+11 2.47e+00 1.43e+03  -1.0 2.82e+00    -  1.62e-03 2.09e-01h
1
  25  4.6836177e+11 2.45e+00 2.94e+03  -1.0 2.08e+00    -  2.59e-01 1.12e-02h
1
  26  5.7869131e+11 2.29e+00 7.73e+03  -1.0 2.06e+00    -  7.98e-01 6.41e-02h
1
  27  5.8156736e+11 2.28e+00 7.72e+03  -1.0 2.24e+00    -  4.38e-03 1.68e-03h
1
  28  7.6671151e+11 2.06e+00 6.75e+03  -1.0 1.95e+00    -  2.64e-03 1.00e-01h
1
  29  7.6904255e+11 2.05e+00 6.77e+03  -1.0 1.83e+00    -  2.24e-02 1.30e-03h
1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr
ls
  30  8.6124739e+11 1.95e+00 6.38e+03  -1.0 1.91e+00    -  1.62e-03 4.96e-02h
1
  31  9.5520746e+11 1.85e+00 6.75e+03  -1.0 1.81e+00    -  6.24e-01 4.98e-02h
1
  32  3.8315751e+12 7.02e-03 1.20e+03  -1.0 1.72e+00    -  1.07e-01 1.00e+00h
1
  33  3.8377240e+12 4.53e-04 1.25e+00  -1.0 4.53e-02    -  1.00e+00 1.00e+00h
1
  34  3.8382084e+12 0.00e+00 2.78e-05  -2.5 4.53e-04    -  1.00e+00 1.00e+00h
1
  35  3.8382075e+12 0.00e+00 8.86e-10  -5.7 1.21e-06    -  1.00e+00 1.00e+00f
1
  36  3.8382075e+12 0.00e+00 3.44e-12  -8.6 7.88e-10    -  1.00e+00 1.00e+00f
1
  37  3.8382075e+12 0.00e+00 4.72e-12 -12.5 1.07e-12    -  1.00e+00 1.00e+00h
1

Number of Iterations....: 37

                                   (scaled)                 (unscaled)
Objective...............:   1.2217393889649695e+05    3.8382074889736309e+12
Dual infeasibility......:   4.7186638881452112e-12    1.4824119805756445e-04
Constraint violation....:   0.0000000000000000e+00    0.0000000000000000e+00
Variable bound violation:   0.0000000000000000e+00    0.0000000000000000e+00
Complementarity.........:   2.8937262932851201e-13    9.0909092644841566e-06
Overall NLP error.......:   1.1390606611048847e-13    1.4824119805756445e-04


Number of objective function evaluations             = 38
Number of objective gradient evaluations             = 38
Number of equality constraint evaluations            = 0
Number of inequality constraint evaluations          = 38
Number of equality constraint Jacobian evaluations   = 0
Number of inequality constraint Jacobian evaluations = 38
```

```
Number of Lagrangian Hessian evaluations             = 37
Total seconds in IPOPT                               = 0.050

EXIT: Optimal Solution Found.
```

In [ ]:
```julia
value.(x)
```

Out[ ]:
```
2-element Vector{Float64}:
 3.3890596192441715
 3.492931776304539
```

In [ ]:
```julia
value.(radius)
```

Out[ ]:
```
1-dimensional DenseAxisArray{Float64,1,...} with index sets:
    Dimension 1, 1:10
And data, a 10-element Vector{Float64}:
 3.4267832172078605
 2.8246876709969095
 1.473929554770824
 2.8538036740546104
 2.537076745951608
 1.6261927152668025
 2.5667015679297327
 2.1963547425501337
 3.0146748153723792
 3.6463692275916513
```

In [ ]: