TD #5: Distance Geometry 2

Large-scale Mathematical Programming

Leo Liberti, CNRS LIX Ecole Polytechnique
`liberti@lix.polytechnique.fr`

INF580

Summary

- Solving an AMPL formulation in python (`amplpy`)
- Solving an SDP formulation in python (`cvxpy`)
- Functions `readDat`, `writeRlz`, `factor`, `MDS`, `PCA`, `mde`, `lde` *given as part of the code for this TD*
- **Task 1**: *test following algs with 2 given DGP instances*
  - SDP+PCA+NLP
  - DDP+PCA+NLP
  - DualDDP+PCA+NLP
- **Task 2**: replace PCA with Barvinok's naive algorithm

Preparing the environment

- Download IPOPT, BonMin, Couenne (and more open-source solvers if you want) by following links in `https://ampl.com/products/solvers/open-source/`
  choose the appropriate architecture (Mac/Lin/Win, choose 64bit unless you have a really old machine)
- Once downloaded, the ipopt, bonmin, couenne executables must be moved to your AMPL directory
- Install Python packages cvxpy, scs, cvxopt, amplpy using `pip install`
  or conda if you use the Anaconda python distribution

amplpy: Running AMPL from Python

- Consider the following test.mod

```
# test.mod
param n integer, > 1;
param m integer, > 1;
set N := 1..n;
set M := 1..m;
param c{N};
param A{M,N};
param b{M};
var x{N} >= 0;
minimize objfun: sum{j in N} c[j]*x[j];
subject to lincon{i in M}: sum{j in N} A[i,j]*x[j] = b[i];
```

- You'll find a corresponding .dat on the course website
- Read model, data, solver, run AMPL, retrieve solution in python

amplpy: Running AMPL from Python

```
from amplpy import AMPL
import numpy as np
lp = AMPL()
lp.read("test.mod")
lp.readData("test.dat")
lp.setOption("solver", "cplex")
lp.solve()
ndata = lp.getData("n")
n = int(ndata.getRowByIndex(0)[0])
solveres = lp.getData("solve_result")
solve_result = solveres.getRowByIndex(0)[0]
objfun = lp.getObjective("objfun")
objfunval = objfun.value()
xvar = lp.getVariable("x")
x = np.zeros(n)
for j in range(n):
    x[j] = xvar[j+1].value()
```

cvxpy: Solving SDPs in Python

```
import sys
import cvxpy as cp
import math
import time
import numpy as np
n = 5
X = cp.Variable((n,n), PSD=True)
A = np.random.rand(n,n)
objfun = cp.trace(A.T*X)
constr1 = [X[i,i+1] + X[i,i+2] <= -1 for i in range(n-2)]
constr2 = [cp.diag(X) == 1]
objective = cp.Minimize(objfun)
constraints = constr1 + constr2
prob = cp.Problem(objective, constraints)
prob.solve(cp.SCS, verbose=True)
Xv = X.value
print(Xv)
```

SDP/DDP/DualDDP + PCA + NLP

- **Task 1**: *test following algs with 2 given DGP instances*
  - SDP+PCA+NLP
  - DDP+PCA+NLP
  - DualDDP+PCA+NLP
- **Task 2**: replace PCA with Barvinok's naive algorithm
- Which is best on quality and efficiency: PCA or Barvinok?