```
n = 100; (*number of nodos*)
maxindegree = 100; (*max in-degree*)
repsp = 10; (*times we use the same probability value p*)

complexity = Table[0, repsp];
(*list to save K-complexity (1-D representation) for the same p value*)
complexitymat = Table[0, repsp];
(*list to save K-complexity (adjacency matrix) for the same p value*)
entropy = Table[0, repsp]; (*list to save entropy values for the same p value*)
ShannonEntropy = Table[0, maxindegree]; (*list to save the entropy for each p value*)
seqnet = Table[0, maxindegree];
(*list to save the K-complexity for each p value (1-D representation)*)
Matnet = Table[0, maxindegree];
(*list to save the K-complexity for each p value (2-D representation)*)

Do[
 Do[
  (*create the random digraph*)
  k = l; (*in-degree*)
  Mnodes = {}; (*adjacency matrix*)
  For[i = 1, i ≤ n, i++×
     AppendTo[Mnodes, Table[0, n]]]×
   For[i = 1, i ≤ n, i++,
    flag = 0;
    While[flag < k, position = RandomInteger[{1, n}];
     If[Mnodes[[i, position]] == 0, Mnodes[[i, position]] = 1; flag++,]]];

  rg = Transpose[Mnodes]; (*random digraph*)
  mat = Flatten[rg]; (*1-D representation of the random graph*)
  If[First[mat] == 0, output = "0" <> ToString[FromDigits[mat]],
   output = ToString[FromDigits[mat]]]; (*transform the list into a string*)

  complexity[[j]] = StringBDM[output];
  complexitymat[[j]] = BDM[rg, 4] // N;
  entropy[[j]] = Entropy[output];
  , {j, repsp}];

     (*perform a trimmed mean discarding the first and fourth quartiles*)
 seqnet[[l]] = TrimmedMean[complexity,
   {(Length[Select[complexity, # < Quantile[complexity, 1/4] &]])/Length[complexity],
    (Length[Select[complexity, # > Quantile[complexity, 3/4] &]])/Length[complexity]}];
 Matnet[[l]] = TrimmedMean[complexitymat,
   {(Length[Select[complexitymat, # < Quantile[complexitymat, 1/4] &]])/Length[complexitymat],
    (Length[Select[complexitymat, # > Quantile[complexitymat, 3/4] &]])/Length[complexitymat]}];
 ShannonEntropy[[l]] = TrimmedMean[entropy,
   {(Length[Select[entropy, # < Quantile[entropy, 1/4] &]])/Length[entropy],
    (Length[Select[entropy, # > Quantile[entropy, 3/4] &]])/Length[entropy]}];
 , {l, 1, maxindegree}]

    (*plot the results*)
ListLinePlot[{Rescale[seqnet, {0, Max[seqnet]}],
  Rescale[Matnet, {0, Max[Matnet]}], Rescale[ShannonEntropy, {0, Max[ShannonEntropy]}]},
 AxesLabel → Automatic, PlotRange → All, PlotLegends →
  Placed[{"K-Complexity (1-D representation)", "K-Complexity (adjacency matrix)", "Entropy"},
   {.5, .15}], Frame → True, GridLines → Automatic, FrameLabel → {"d⁻
```