```
n = 4; (*number of functions in each set*)
k = 2 (*parameter k*) numexp = 40 000; (*number of sets to generate*)
numperms = 16; (*number of permutations to test*)

complexity = Table[0, numexp]; (*list for K-complexity (1-D representation)*)
complexitymat = Table[0, numexp]; (*list for K-complexity (2-D representation)*)
entropy = Table[0, numexp]; (*list for entropy values*)

Do[booleanfunction = {};
 Matrixbooleanfunctionlist = {}; (*matrix representation*) Clear[a];
 A = Table[a[i], {i, 1, k}]; (*boolean variables*)
     (*generate the set of boolean functions*)
 functions = Table[0, n];
 For[i = 0, i < n, i++,
  f = BooleanFunction[RandomInteger[{0, (2^2^k - 1)}], k];
  (*choose randomly one of the 2^2^k possible boolean functions*)
  AppendTo[booleanfunction, f];
  AppendTo[Matrixbooleanfunctionlist,
   Boole[BooleanTable[BooleanConvert[Apply[f, A], "NOR"], A]]];
  functions[[i + 1]] = f];

     (*generate the isomorphisms*)
 matrixrepresen = Matrixbooleanfunctionlist;
 perms = RandomChoice[Permutations[Table[i, {i, n}]], numperms];
 isos = DeleteDuplicates[Permute[matrixrepresen, #] & /@ perms];
 complexityiso = Table[0, Length[isos]];
 complexitymatiso = Table[0, Length[isos]];
 entropyiso = Table[0, Length[isos]];
     (*measure the complexity of the isomorphisms*)
 Do[flatten = Flatten[isos[[L]]];
  If[First[flatten] == 0,
   output = "0" <> ToString[FromDigits[flatten]], output = ToString[FromDigits[flatten]]];
  complexityiso[[L]] = StringBDM[output];
  complexitymatiso[[L]] = BDM[isos[[L]], 4] // N;
  entropyiso[[L]] = Entropy[output], {L, Length[isos]}];
     (*the true complexity value is the minimum value obtained from the isos*)
 complexity[[l]] = {l, Min[complexityiso]};
 complexitymat[[l]] = {l, Min[complexitymatiso]};
 entropy[[l]] = {l, Min[entropyiso]};
 Export[NotebookDirectory[] <> ToString[l] <> "file_name.txt", Matrixbooleanfunctionlist],
 {l, 1, numexp}]

    (*sort the results in increasing complexity order*)
sorted = Sort[complexity, #1[[2]] < #2[[2]] &]
list = Flatten[First[sorted[[#]]] & /@ Table[i, {i, Length[sorted]}]];
data = Flatten[Take[sorted[[#]], {2, 2}] & /@ Table[i, {i, Length[sorted]}]];
sortedmat = Sort[complexitymat, #1[[2]] < #2[[2]] &]
datamat = Flatten[Take[sortedmat[[#]], {2, 2}] & /@ Table[i, {i, Length[sortedmat]}]];
sortedent = Sort[entropy, #1[[2]] < #2[[2]] &] // N
dataent = Flatten[Take[sortedent[[#]], {2, 2}] & /@ Table[i, {i, Length[sortedent]}]];

    (*plot the sorted results*)
ListLinePlot[{Rescale[data, {0, Max[data]}], Rescale[datamat, {0, Max[datamat]}],
  Rescale[dataent, {0, Max[dataent]}]}, AxesLabel → Automatic, PlotRange → All,
 PlotLegends → Placed[{"K-Complexity (1-D)", "K-Complexity (2-D)", "Entropy"}, {0.75, 0.3}],
 Frame → True, GridLines → Automatic, FrameLabel → {"Ordered Sets", "C(f)"}]

    (*plot some sets in increasing complexity order*)
numsets = 16; (*number of sets to visualize*)
r = 1; sets = {};
Table[If[nset == Round[(numexp * r / numsets)] || nset == 1, r++;
   statemat =
    ReadList[NotebookDirectory[] <> ToString[list[[nset]]] <> "file_name.txt", Expression];
   AppendTo[sets, list[[nset]]];
   Show[MatrixPlot[statemat, ColorFunction → "Monochrome"]]
  ], {nset, 1, numexp}] /. Null → Sequence[]
Print["sets by increasing complexity order = ", sets]
```