This is the Online Resource 1 attached to the paper:
"*A 99 line code for discretized Michell truss optimization written in Mathematica*"
submited to Structural and Multidisciplinary Optimization
Author: *Tomasz Sokol*, t.sokol@il.pw.edu.pl
Department of Structural Mechanics and Computer Aided Engineering,
Faculty of Civil Engineering, Warsaw University of Technology,
Al. Armii Ludowej 16, 00-637 Warsaw, Poland

Program - definitions

Patterns of elements

```
Needs["Developer`"];

GenInc[dist_] := ToPackedArray[Flatten[Reap[
      Sow[{{1, 0}, {0, 1}, {1, 1}, {-1, 1}}];
      Do[If[GCD[i, j] == 1,
         Sow[{{i, j}, {-i, j}, {j, i}, {-j, i}}]]
       , {i, 2, dist}, {j, 1, i - 1}]][[2]], 2]]

Patterns[nx_, ny_, distx_, disty_] := Module[
  {idx, idy, i0, i1, ne, INC},
  INC = GenInc[Max[distx, disty]];
  ToPackedArray[Reap[Do[
       {idx, idy} = INC[[i]];
       {i0, i1} = If[idx ≥ 0, {0, nx - idx}, {-idx, nx}];
       If[i0 ≤ i1 && idy ≤ ny &&
          Abs[idx] ≤ distx && idy ≤ disty,
        Sow[{idx, idy, i0, i1, (i1 - i0 + 1) (ny - idy + 1)}]]
       , {i, Length[INC]}]][[2, 1]], Integer]]
```

Lengths and directional cosines

```
LCS[patt_, DX_, DY_] := Module[{L, c, s},
  ToPackedArray[Table[
    c = patt[[p, 1]] DX; s = patt[[p, 2]] DY;
    L = Sqrt[c^2 + s^2]; c /= L; s /= L;
    {L, c, s}
    , {p, Length[patt]}], Real]]

VectorL[patt_, lcs_] := Module[{ne, L},
  ToPackedArray[Flatten[Table[
     ne = patt[[p, 5]]; L = lcs[[p, 1]]; Table[L, {ne}]
     , {p, Length[patt]}]], Real]]
```

Elements

```
ElXY[nx_, ny_, patt_, DX_, DY_] := Module[
  {idx, idy, i0, i1, ne, X, Y},
  X = ToPackedArray[Table[i DX, {i, 0, nx}], Real];
  Y = ToPackedArray[Table[j DY, {j, 0, ny}], Real];
  ToPackedArray[Flatten[Reap[
      Do[{idx, idy, i0, i1, ne} = patt[[p]];
       Sow[Table[{{X[[i]], Y[[j]]}, {X[[i + idx]], Y[[j + idy]]}}
          , {j, 1, ny - idy + 1}, {i, i0 + 1, i1 + 1}]];
       , {p, Length[patt]}]][[2]], 3], Real]]
```

Boundary conditions

```
BCList[supports_, nx_] := Module[{k, i, j, ux, uy},
  ToPackedArray[Partition[Union[Flatten[Reap[
         Do[{{i, j}, {ux, uy}} = supports[[s]];
          k = 2 ((nx + 1) j + i) + 1;
          If[ux == 1, Sow[k]]; If[uy == 1, Sow[k + 1]];
          , {s, Length[supports]}]][[2, 1]]], 1], Integer]]
```

Matrix B^T

```
MatrixBT[nx_, ny_, patt_, lcs_] := Module[
  {idx, idy, i0, i1, L, c, s, rules, k, dk, ie = 0},
  rules = Reap[Do[{idx, idy, i0, i1, k} = patt[[p]];
        {L, c, s} = lcs[[p]];
        dk = 2 ((nx + 1) idy + idx);
        Do[ie++; k = 2 ((nx + 1) j + i) + 1;
         Sow[{ie, k} → -c]; Sow[{ie, k + 1} → -s];
         k += dk;
         Sow[{ie, k} → c]; Sow[{ie, k + 1} → s];
         , {j, 0, ny - idy}, {i, i0, i1}];
        , {p, Length[patt]}];][[2, 1]];
  Transpose[SparseArray[rules]]]

MatrixBT[nx_, ny_, patt_, lcs_, BC_] :=
 Delete[MatrixBT[nx, ny, patt, lcs], BC]
```

Loading vector

```
VectorP[loads_, nx_, ny_] := Module[{P, i, j, n, p},
  P = Table[{0, 0}, {(nx + 1) (ny + 1)}];
  Do[{{i, j}, p} = loads[[f]];
   n = (nx + 1) j + i + 1; P[[n]] = p;
   , {f, Length[loads]}];
  ToPackedArray[Flatten[P], Real]]

VectorP[loads_, nx_, ny_, BC_] := ToPackedArray[
  Delete[VectorP[loads, nx, ny], BC], Real]
```

■ Solution of the optimization problem

```
OptimalTruss[xmax_, ymax_, nx_, ny_, supports_, loads_,
  distx_: 1, disty_: 0, kappa_: 1, tol_: Sqrt[$MachineEpsilon]] :=
 Module[{ndx, ndy, dx, dy, pat, lcs, L, LL, BC, BB, PP,
   nn, ne, dof, S, A, e, g, a, t, vol, P, B},

  ndx = Min[nx, Max[1, distx]];
  ndy = Min[ny, Max[1, If[disty < 1, distx, disty]]];
  dx = xmax / nx; dy = ymax / ny;
  pat = Patterns[nx, ny, ndx, ndy];
  nn = (nx + 1) (ny + 1);
  ne = Total[pat[[All, 5]]];
  BC = BCList[supports, nx];
  dof = 2 nn - Length[BC];
  Print["Mesh ", nx, "x", ny, ":", ndx, "x", ndy,
   ",  Nodes ", nn, ",  Elements ", ne, ",  DOF ", dof];
  lcs = LCS[pat, dx, dy];
  L = VectorL[pat, lcs] ;
  P = VectorP[loads, nx, ny, BC];
  B = MatrixBT[nx, ny, pat, lcs, BC];

  PP = Transpose[{P, Table[0, {Length[P]}]}];
  LL = Join[L, kappa L];
  BB = Join[B, -B, 2];
  Print["Matrix H ", Length[P], "x", Length[LL], " in ",
   ByteCount[BB] / 2.^20, "MB (", 16 dof ne / 2.^30, "GB full)"];
  t = Timing[S = LinearProgramming[LL, BB, PP,
        Method → "InteriorPoint", Tolerance → tol];][[1]];
  vol = S.LL;
  Print["Objective S.L = ", vol, "  CPU time = ", t, "s"];
  S = Partition[S, ne];
  S = S[[1]] - S[[2]];
  A = Table[If[S[[i]] < 0, -kappa S[[i]], S[[i]]], {i, ne}];
  A /= Max[A];
  A = Chop[A, 100 tol];

  e = ElXY[nx, ny, pat, dx, dy];
  Graphics[Reap[Do[a = A[[i]]; If[a > tol,
        Sow[{Thickness[.015 Sqrt[a]], Hue[.7 (1 - a)],
          Line[e[[i]]]}]], {i, ne}]][[2, 1]]]
```

Example

```
Xmax = 3; Ymax = 1;
supports := Table[{{0, i}, {1, 1}}, {i, 0, NY}];
loads := {{{NX, Round[NY / 2]}, {0, -1}}};
NY = 10; NX = 3 NY; DIST = 10;
OptimalTruss[Xmax, Ymax, NX, NY, supports, loads, DIST]
```

```
Mesh 30x10:10x10,  Nodes 341,  Elements 19632,  DOF 660
Matrix H 660x39264 in 1.73669MB (0.193076GB full)
Objective S.L = 13.6857  CPU time = 1.969s
```