Euclidean semicoupling problem.

Before realizing our inability to uniformly sample small local CAT(0)-neighborhoods (i.e. small ball in Poincare's disk), we were looking to solve an analogous (but qualitatively very distinct) semicoupling problem in the euclidean plane. Below we present our basic computations for this euclidean test case, and describe our cost functions. Our refusal to use *Mathematica*'s secretive inbuilt `ImplicitRegion' function and their constrained minimization functions means, we must do everything ourself, and use only the MinimalBy function.

Outline of the computations below: (i) first we construct the sample source and boundary target space using hexagonal lattice approximation in the plane. Letting quad denote the quadratic geodesic function on euclidean plane, i.e. quad[x,y] equals the |x-y|^2/2 in common notation, we define a cost pairing between source and target via the formula: cost[y0, x0] = \sup$_{Y\,\text{boundary}}$ t.quad[y0, y'] – quad[x0, y'],where t is a real positive parameter and approximately equal to zero. Having defined the cost pairing, one must then be free to compute c-Legendre-Fenchel transforms for potentials \$$\phi$:Y \to R\$ or \$$\psi$:X\to R\$. (i.e. potentials on the boundary and source). The basic fact of optimal transportation is that optimal couplings are supported on the graphs of \$c\$-subdifferentials of \$c\$-convex/concave potentials. We recall the basic definitions ϕ^c(x0) = \sup$_{y'\text{:boundary}}$ ϕ(y') – cost(y', x0) and ϕ^c(y0) = \inf$_{x'\text{:source}}$ ψ(x') + cost(y0, x'). A function ϕ defined on the boundary target space is \$c\$-concave if $\phi^{cc} = \phi$. In this case, the value of ϕ(y0) coincides with the infimum of $\phi^c + c_{y0} = \phi^c(x')$+c(y0,x') on the source. The source points $x^\#$at which the infimum is attained constitute the \$c\$-subdifferential $\delta\phi$(y0). The mapping y' -> $\delta\phi$(y') realizes an optimal semicoupling between the boundary target space and the corresponding activated target space, here the activated source mass being described by the image of the subdifferential mapping.

Optimal transports are characterized by being supported on the graphs of c-concave potentials. Every such potential defines an optimal transport from the boundary measure to the source. Our difficulty is to identify concave potentials whose subdifferentials pushforward the boundary measure to the uniform source mass. We have not(!) succeeded in this computation.

```
hex[m_Integer] := hex[m] =
   N@{{1/n, 0}, {-1/n, 0}, {-1/(2 n), Sqrt[3]/(2 n)}, {1/(2 n), -Sqrt[3]/(2 n)},
      {1/(2 n), Sqrt[3]/(2 n)}, {-1/(2 n), -Sqrt[3]/(2 n)}, {0, 0}} /. n → 2^m;

gen[z_List, m_Integer: 8] :=
  gen[z, m] = Table[hex[m][[j]] + #, {j, 7}] & /@ z // Union // Flatten[#, 1] &;
cgen = Compile[{z, m}, gen[z, m]];

hexgen[z_List, g_Integer: 1, m_Integer: 8] :=
  hexgen[z, g, m] = Nest[Union@cgen[#, m] &, z, g];

quad[x_List, y_List] := quad[x, y] = N@((x[[1]] - y[[1]])^2 + (x[[2]] - y[[2]])^2)/2;
```

The hexgen function grows a g-generational hexagonal lattice from within a g.2^{-m}-neighborhood of the list z.
(I dont understand the content of the following error message.)

```
hexgen[{{0, 0}}, 4, 2] // ListPlot
```

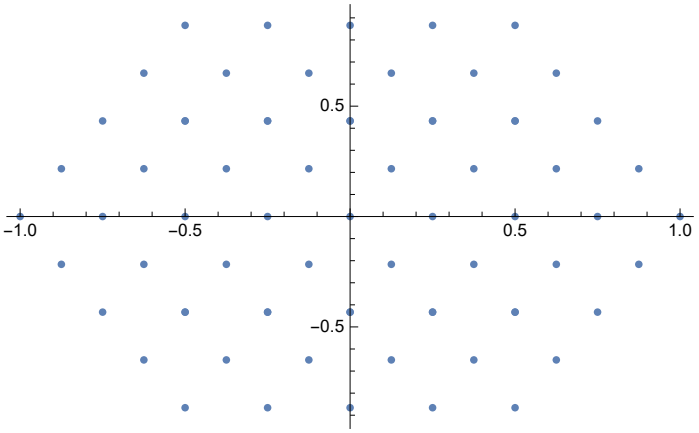
CompiledFunction::cfsa : Argument {{0, 0}} at position 1 should be a machine–size real number. ≫

CompiledFunction::cfsa :

Argument $\left\{\{-0.25, 0.\}, \{-0.125, -0.216506\}, \{-0.125, 0.216506\}, \{0., 0.\}, \{0.125, -0.216506\}, \{0.125, 0.216506\}, \left\{\frac{1}{4}, 0.\right\}\right\}$ at position 1 should be a machine–size real number. ≫

CompiledFunction::cfsa : Argument $\left\{\{-0.5, 0.\}, \{-0.375, -0.216506\}, \{-0.375, 0.216506\}, \{-0.25, -0.433013\}, \{-0.25, 0.\}, \{-0.25, 0.433013\}, \{-0.125, -0.216506\}, \{-0.125, 0.216506\}, \{0., -0.433013\}, \{0., 0.\}, \{0., 0.433013\}, \{0.125, -0.216506\}, \{0.125, 0.216506\}, \{0.25, -0.433013\}, \{0.25, 0.\}, \{0.25, 0.433013\}, \{0.375, -0.216506\}, \{0.375, 0.216506\}, \left\{\frac{1}{2}, 0.\right\}\right\}$ at position 1 should be a machine–size real number. ≫

General::stop : Further output of CompiledFunction::cfsa will be suppressed during this calculation. ≫



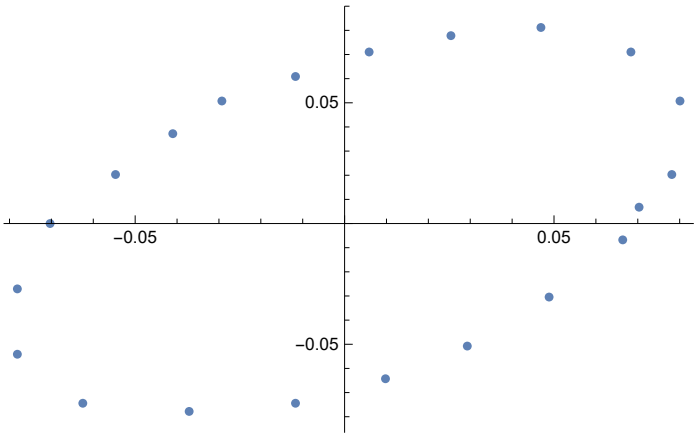
```
disk1 = Union[hexgen[{{0, 0}}, 30, 8] /.
    {x1_, x2_} /; x1^2 - x1 x2 + x2^2 > 0.004 + 0.001 -> Nothing];
boundary1 = hexgen[{{0, 0}}, 30, 8] /. {x1_, x2_} /;
     (0.0045 > x1^2 - x1 x2 + x2^2 || x1^2 - x1 x2 + x2^2 > 0.004 + 0.001) -> Nothing;

b[1] = {-0.0546875`, 0.020297470401197778`};
b[2] = {0.080078125`, 0.05074367600299446`};
b[3] = {0.029296875`, -0.05074367600299445`};

Do[
  b[j + 1] =
   MaximalBy[boundary1, Min[Table[quad[b[k], #], {k, j}]] &] // RandomChoice,
  {j, 3, 20}];
ListPlot[Table[b[d], {d, 21}]] // Print;

quad[x_List, y_List] :=
  quad[x, y] = N@((x[[1]] - y[[1]])^2 + (x[[2]] - y[[2]])^2)/2;
```

CompiledFunction::cfsa : Argument {{0, 0}} at position 1 should be a machine–size real number. ≫

CompiledFunction::cfsa : Argument
{{−0.00390625, 0.}, {−0.00195313, −0.00338291}, {−0.00195313, 0.00338291}, {0., 0.}, {0.00195313, −0.00338291}, {0.00195313, 0.00338291}, $\left\{\frac{1}{256}, 0.\right\}$} at position 1 should be a machine–size real number. ≫

CompiledFunction::cfsa : Argument
{{−0.0078125, 0.}, {−0.00585938, −0.00338291}, {−0.00585938, 0.00338291}, {−0.00390625, −0.00676582}, {−0.00390625, 0.}, {−0.00390625, 0.00676582}, {−0.00195313, −0.00338291}, {−0.00195313, 0.00338291}, {0., −0.00676582}, {0., 0.}, {0., 0.00676582}, {0.00195313, −0.00338291}, {0.00195313, 0.00338291}, {0.00390625, −0.00676582}, {0.00390625, 0.}, {0.00390625, 0.00676582}, {0.00585938, −0.00338291}, {0.00585938, 0.00338291}, $\left\{\frac{1}{128}, 0.\right\}$}
at position 1 should be a machine–size real number. ≫

General::stop : Further output of CompiledFunction::cfsa will be suppressed during this calculation. ≫



Below cost1 is our nonstandard cost pairing between the boundary and target space.

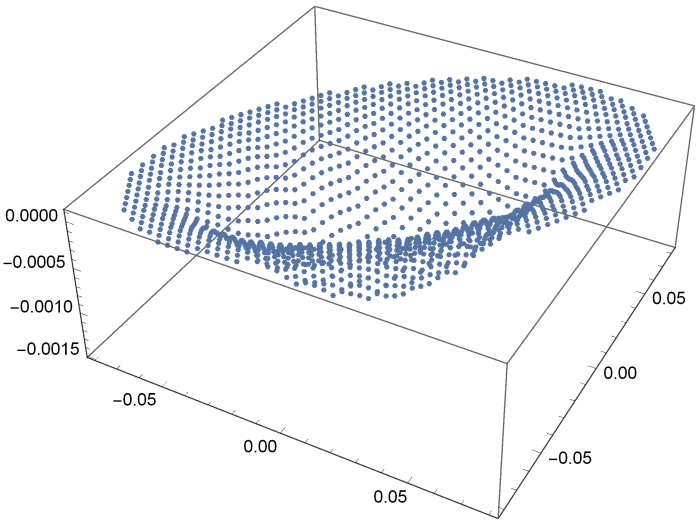
```
cost1[y0_List, x0_List, t_Real: 0.01] :=
  cost1[y0, x0, t] = t quad[y0, #] - quad[x0, #] &@
    MaximalBy[boundary1, N@(t quad[y0, #] - quad[x0, #]) &][[1]];

ccost1 = Compile[{y, x, t}, cost1[y, x, t]];
Append[#, ccost1[b[1], #, 0.01]] & /@ disk1 // ListPointPlot3D
```

CompiledFunction::cfsa : Argument {−0.0546875, 0.0202975} at position 1 should be a machine−size real number. ≫

CompiledFunction::cfsa : Argument {−0.0546875, 0.0202975} at position 1 should be a machine−size real number. ≫

CompiledFunction::cfsa : Argument {−0.0546875, 0.0202975} at position 1 should be a machine−size real number. ≫

General::stop : Further output of CompiledFunction::cfsa will be suppressed during this calculation. ≫



**Given a target boundary measure,
the Kantorovich duality theory of optimal transport
compels us to construct c – concave boundary potentials,
compute their subdifferentials and their mass.**

We have been ineffective in completing these steps via Mathematica. WIthin theory, the process for constructing an optimal semicoupling is somehow ' obvious', namely, "fill from minimal sublevel sets". The implementation is difficult.

Our underlying motivation for studying this optimal semicoupling problem is to experimentally verify that the singularity structure of an optimal semicoupling has a consistent topology. I.e., if we perturb the boundary target measure while fixing the uniform source measure per our nonstandard cost1, then when the source mass is sufficiently near the target mass $\sigma[X] > \tau[Y]$, then the singularity structure, i.e. source points x' where the subdifferential of the c-convex source potential ψ is multivalued, has a consistent topology. I.e., varying the target or source measure induces continuous homotopies between the correspondant underlying singularities.

An optimal semicoupling is a measure π supported on a c-cyclically monotone subsets of the product $Y \times X$. Effectively, this means that the crosscost function cross : $S_*(Y\times X) \to R$ which is defined on the unordered configuration space of Y\times X must be $>=0$ on configurations supported on the support of π. We attempted a gradient scheme to construct such positive-valued sets. This procedure was hopeless, however. It scales exponentially with the number of points in the configuration.

```
cross[y_List, x_List, a_List: {2, 3, 4, 1}] := cross[y, x, a] =
   N@Sum[ccost1[y[[j]], x[[j]], 0.01] - ccost1[y[[j]], x[[a[[j]]]], 0.01], {j, 4}];
sigma = {2, 3, 4, 1}; s[j_Integer] := s[j] = s[j - 1][[sigma]]; s[1] = sigma;
crossvec[u_List, v_List] :=
  crossvec[u, v] = (Table[cross[u, #, s[j]], {j, 3}] & /@ {v})[[1]];


ub = {{-0.078125`, -0.027063293868263706`}, {0.0546875`, -0.02029747040119778`},
   {-0.064453125`, 0.010148735200598885`}, {-0.0625`, 0.013531646934131857`}};

ccrossvec_ub = Compile[{v}, crossvec[ub, v]];
cmono = Compile[{x}, AnyTrue[{1, 2, 3}, crossvec[ub, x][[#]] > 0 &]];

ccomp = Compile[{m[j]}, Tuples@Table[chexgen[{m[j][[k]]}, 1, 8], {k, 4}] /.
    z_List /; cmono[z] :> Nothing];
```