Convex Cover

Given a irregular, closed, convex polygon $\mathscr{P}$ with $n-1$ sides and m circle-centers $\{(x_i, y_i)\}_i^m$ contained within that polygon, compute the radii, $0 \leq r_i$, of m circles centered at those m points such that the sum of the areas of the circles is minimized (approximately) and that any vertex in $\mathscr{P}$ is also contained in at least one of the m circles.

Here is the function signature: `find_convex_cover(pvertices,clist)` where `pvertices` is a $(n-1)$-long iterable of polygon vertices and `clist` is a list of (x_i, y_i) tuples of circle-centers. The output of `find_convex_cover` is a m long list of radii, r_i, corresponding to the m circle-centers.

just Heuristic is enough

Example:

```
>>> pvertices = array([[ 0.573,  0.797],
                       [ 0.688,  0.402],
                       [ 0.747,  0.238],
                       [ 0.802,  0.426],
                       [ 0.757,  0.796],
                       [ 0.589,  0.811]])

>>> clist = [(0.7490863467660889, 0.4917635308023209),
             (0.6814339441396109, 0.6199470305156477),
             (0.7241617773773865, 0.6982813914515696),
             (0.6600700275207232, 0.7516911829987891),
             (0.6315848053622062, 0.7730550996176769),
             (0.7348437356868305, 0.41342916986639894),
             (0.7597683050755328, 0.31729154508140384)]

>>> find_convex_cover(pvertices,clist) # note some radii == 0
[0, 0, 0.10297280518543134, 0, 0.06374182913818943, 0.0684588720095565, 0.07987784828713643]
```

Hints:

- m can be very large so use Numpy broadcasting effectively.
- For your own understanding, use Matplotlib to visualize the polygons and circles.
- Numpy is the only third-party module you can use with this assignment.
- Since the n-polygon is closed, the first and last vertices are the same so that only $n-1$ vertices need be specified.
- Your solution can be an approximation to the minimum.

Please put your Python code in a Python script file and upload it. Please retain your submitted source files! Remember to use all the best practices we discussed in class. You can use any module in the Python standard library, but third-party modules (e.g., Numpy, Pandas) are restricted to those **explicitly** mentioned in the problem description.

Tips:

- After you have submitted your file, do **not** use the browser back or reload buttons to navigate or open the page in multiple browser tabs, as this may cause your `attempts` to decrease unexpectedly. It may take up to thirty seconds for your code to be processed, so please be **patient**.
- If you find yourself back at the main page without any feedback or change in your `attempts` then it means that your code timed out or crashed in some unexpected way.
- Ensure that your development environment does not presume the existence of certain packages for the autograder. The autograder does not have anything other than the standard library and those third-party libraries **explicitly** named in the problem description.
- Do not leave extraneous statements in your code like test cases, print statements, or anything else besides what is needed to evaluate your submission because the the autograder will spend its limited time executing those lines, which may result in unexpected crashes or timeouts.

浏览... 未选择文件。 Upload Python source code file