The Waiter Problem

Problem Statement

Placing glasses on a tray. You serve drinks in a restaurant and must carry a tray with one hand. As you load glasses onto the tray, the tray can become unstable – your hand must stay under the center of mass of the glasses. There are particular spots on the tray where the glasses must be placed, but you are free to place the glasses in any order you want. How should you do it to make sure the tray stays balanced? Let's model this problem as follows, thinking of the glasses as "points" in the plane. Let S = {p1, . . . , pn} be a set of n points in the plane (e.g., entered by mouse clicks or randomly generated). Let c(X) denote the center of mass of the set X ⊆ S of points. Our goal is to find a "good" ordering (permutation π) of the points S, ($p_{\pi 1}$, $p_{\pi 2}$, . . . , $p_{\pi n}$), so that the center of mass, cj = c({$p_{\pi 1}$, . . . , $p_{\pi j}$ }), of the first j points (j = 1, 2, . . . , n) in the order does not "move around" too much. The "score" for a given ordering π might be (a) the smallest radius r of a disk centered at cn = c(S) that contains all points cj ; (b) the area of the convex hull of the centers of mass cj (for j = 1, 2, . . . , n); or (c) the total length of the chain (c1, c2, c3, . . . , cn) (i.e., the total distance traveled by the center of mass as the points were added in the order π.

Approach

The motive of this project was to try different heuristics and analyze how does each standout with respect to different scoring mechanism.

Code

A heuristic needs to give the next point at each step of the algorithm for j = 1..k, given points 1..n, where k can be as large as n. The structure of heuristic API is like below:

```
public abstract Point getNextPoint(List<Point> points);
```

A scoring metric tells us, what is the current score as we added j out of n points with respect to certain criteria. At each step of the algorithm, we process the point and update the score. The structure of API looks like below:

```
public abstract float getScore();

public abstract void process(Point point);
```

Algo : Input : set of points P, n, k

```
P' = P; O = {}
while( k > 0 )
   Point p = heuristic.getNextPoint(P');
   score.process(p);
   O = O + p; P' = P' \ p; k = k - 1;
} return score.getScore();
```

Heuristics

- Minimum Distance from Fixed COM (MinDistanceFixedCOM) : Next point is chosen the one with minimum distance from the COM of all n points.
- Closest Distance from Variable COM (ClosestDistanceVariableCOM) : Next point is chosen the one with minimum distance from COM of the added j points. The first point is added which has minimum distance from the COM of all n points.
- Minimize COM Variance (MinimizeCOMVariance) : Next point is chosen based on minimum distance between COM of first j points and COM of all n points.

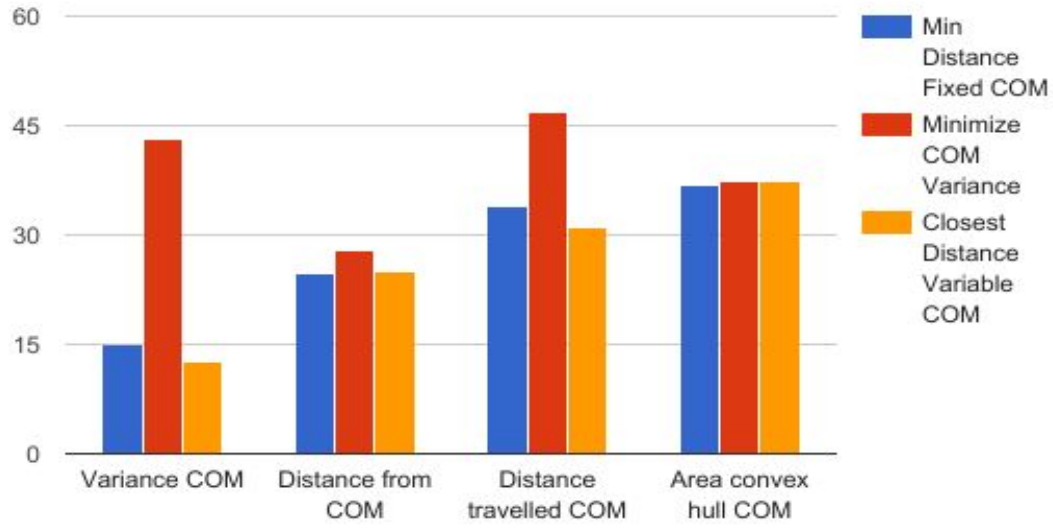
Score

- Smallest radius which contains all the generated COM of first j points added with COM of n as center (DistFromCOM).
- Area of convex hull of generated COM (AreaConvexHullCOM)
- Distance travelled by generated COM (DistTravelledByCOM)
- Variance in COM with respect to COM of all n points (VarianceCOM)

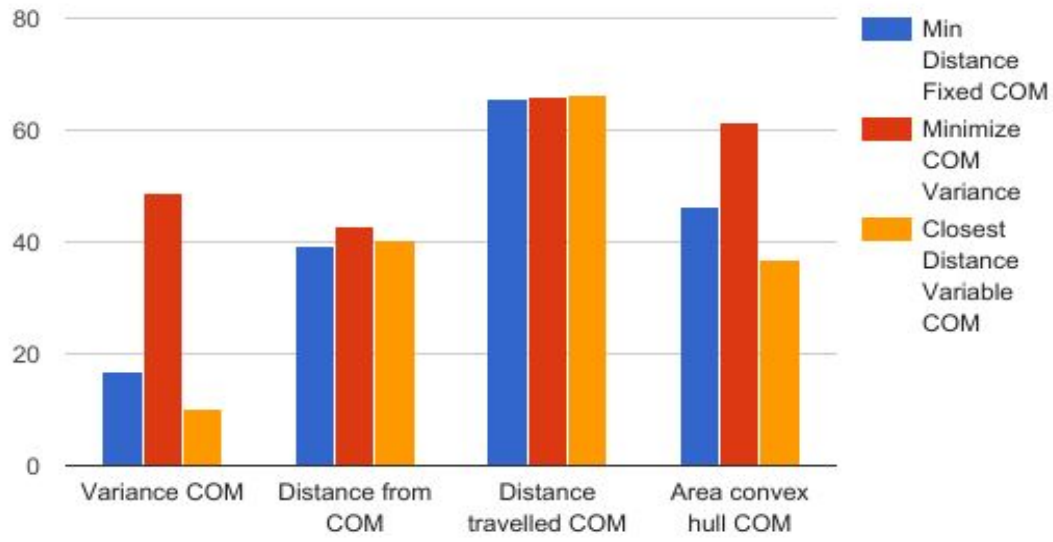
Results

The following results were obtained from the study.

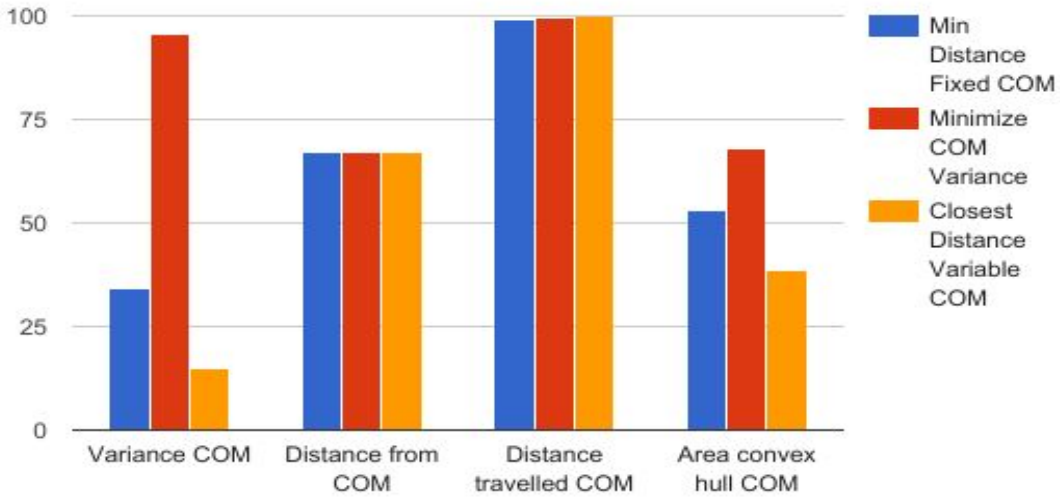
1. When adding 20 points when 100 random points are given for 5000 iterations.



2. When adding 50 points when 100 random points are given for 5000 iterations.



3. When adding 100 points when 100 random points are given for 5000 iterations.



Result

For almost each case, Minimizing distance from fixed center of mass gave better results for all the scoring metric.