Clustering with Neighborhoods is Hard for Squares*

Georgiy Klimenko† Benjamin Raichel‡

Abstract

In the clustering with neighborhoods problem one is given a set $\mathcal{S}$ of disjoint convex objects in the plane and an integer parameter $k \geq 0$, and the goal is to select a set C of k center points in the plane so as to minimize the maximum distance of an object in $\mathcal{S}$ to its nearest center in C. Previously [HKR21] showed that this problem cannot be approximated within any factor when $\mathcal{S}$ is a set of disjoint line segments, however, when $\mathcal{S}$ is a set of disjoint disks there is a roughly 8.46 approximation algorithm and a roughly 6.99 approximation lower bound. In this paper we investigate this significant discrepancy in hardness between these shapes. Specifically, we show that when $\mathcal{S}$ is a set of axis aligned squares of the same size, the problem again is hard to approximate within any factor. This surprising fact shows that the discrepancy is not due to the fatness of the object class, as one might otherwise naturally suspect.

1 Introduction

Given a set of n points P in a metric space and an integer parameter $k \geq 0$, in the standard k-center clustering problem the goal is to select a set C of k center points from the metric space (or in the discrete variant $C \subseteq P$) so as to minimize the maximum distance of a point in P from its nearest center in C. This fundamental problem and its variations have been well studied in the computational geometry community. For the standard problem there is a well known greedy 2-approximation algorithm due to Gonzalez [Gon85], and an iterative scooping based 2-approximation algorithm due to Hochbaum and Shmoys [HS85]. Conversely, for general metric spaces it is NP-hard to approximate within any factor less than 2, and even for points in the plane the problem remains hard to approximate within a factor of roughly 1.82 [FG88].

While many variants of k-center clustering have been considered, here we focus on the problem of *k-center clustering with neighborhoods* introduced recently in [HKR21]. In this problem the input is a set $\mathcal{S}$ of n disjoint convex objects in the plane, and the goal is again to select a set C of k points from the plane so as to minimize the maximum distance of an object in $\mathcal{S}$ from its nearest center in C. Note that the standard k-center problem is a special case of k-center clustering with neighborhoods where $\mathcal{S} = P$ is a discrete point set.

In [HKR21] it was shown that clustering with neighborhoods is hard to approximate within any factor when $\mathcal{S}$ is a set of disjoint segments. Conversely, it was also shown that when $\mathcal{S}$ is a set of disjoint disks the problem is $\frac{\sqrt{13}-\sqrt{3}}{2-\sqrt{3}} \approx 6.99$ hard to approximate, and additionally a near matching $(5 + 2\sqrt{3}) \approx 8.46$ approximation algorithm was given. In other words, for disks the problem is APX-complete.

The clustering with neighborhoods problem can be equivalently defined as finding k equal radius balls of the smallest possible radius such that every object has non-empty intersection with at least one of the balls. Alternatively, one could require that each object is entirely contained in one of the balls. This however, implies that the optimal radius is at least the radius of the largest object, whereas in our case the optimal radius can be arbitrarily smaller. This significantly and provably changes the hardness of the two problems. Specifically, Xu and Xu [XX10] considered k-center clustering on point sets where given points sets $S_1, \ldots, S_n$ the goal is to find k balls of minimum radius such that each S_i is entirely contained in one of the balls. For this problem they achieved a $(1 + \sqrt{3})$-approximation, whereas clustering with neighborhoods cannot in general be approximated within any factor in polynomial time unless $\mathsf{P} = \mathsf{NP}$.

Motivation and Contribution. As discussed above, clustering with neighborhoods is hard to approximate within any factor when the objects are disjoint line segments, however, when the objects are disks there is a constant factor approximation. This intriguingly large hardness gap between segments and disks begs the question, what geometric feature accounts for this gap? One may naturally suspect (as the authors did), that the difference is due to fatness, as segments are arbitrarily skinny objects whereas disks are fat. It is well known

*A preliminary version of the main proof in this paper appeared in Georgiy Klimenko's thesis [Kli23]

†Department of Computer Science; University of Texas at Dallas; Richardson, TX 75080, USA; gik140030@utdallas.edu; Work on this paper was partially supported by a NSF CAREER Award 1750780.

‡Department of Computer Science; University of Texas at Dallas; Richardson, TX 75080, USA; benjamin.raichel@utdallas.edu; http://utdallas.edu/~benjamin.raichel. Work on this paper was partially supported by a NSF CAREER Award 1750780.

that this basic geometric property can often make a significant difference in the difficulty of a problem (e.g. [Cha03]). Surprisingly, however, in this paper we show that when the objects are disjoint squares (one of the simplest classes of fat objects), not only does the constant factor approximation algorithm break down, but in fact the problem is again hard to approximate within any factor, as was the case for line segments. Moreover, we show this is true even when the squares are axis aligned and all of equal size. Indeed, this paper shows the hardness gap is not due fatness, but rather roughly speaking concerns more how pointed the objects are. (More precisely, it concerns how closely one can place three disjoint objects to a single point.)

2 Preliminaries

Given points $x, y \in \mathbb{R}^d$, $||x-y||$ denotes their Euclidean distance. Given two closed sets $X, Y \subset \mathbb{R}^d$, $||X-Y|| = \min_{x\in X, y\in Y} ||x-y||$ denotes their distance. For a point x and a value $r \geq 0$, let $B(x,r)$ denote the closed ball centered at x and with radius r. [HKR21] considered the following problem.

Problem 1 (Clustering with Neighborhoods) *Given a set $\mathcal{S}$ of n disjoint convex objects in the plane, and an integer parameter $k \geq 0$, find a set of k points C (called centers) which minimize the maximum distance to a convex object in $\mathcal{S}$. That is,*

$$C = \arg \min_{C'\subset\mathbb{R}^2, |C'|=k} \max_{S\in\mathcal{S}} ||S - C'||.$$

Let C be any set of k points, and let $r = \max_{S\in\mathcal{S}} ||S-C||$. We refer to r as the *radius* of the solution C, since r is the minimum radius such that the set of all balls $B(c,r)$ for $c \in C$, intersect all $S \in \mathcal{S}$. If C is an optimal solution then we refer to its radius r_{opt} as the optimal radius. Let $\mathcal{S}, k$ be an instance of Problem 1 with optimal radius r_{opt}. For a value $\alpha \geq 1$, we refer to a polynomial time algorithm as an α-approximation algorithm if it returns a solution C of size k such that the radius is $\leq \alpha r_{opt}$.

3 Hardness for Squares

In this section we argue that it is hard to approximate Problem 1 within any factor when $\mathcal{S}$ is a set of axis aligned squares of the same size. Our hardness results use a construction similar to the one from [FG88], where they reduce from the problem of planar vertex cover where the maximum degree of a vertex is three. This problem is known to be NP-complete [GJ77], and we denote this problem as P3VC. We remark that the high level approach of reducing from P3VC used in [FG88] has inspired many other hardness reduction for geometric problems, including the prior reductions for clustering with neighborhoods for the cases of segments and disks [HKR21].

Let G, k be an instance of P3VC, and consider a straight line embedding of G. In particular, in $O(n \log n)$ time one can compute an straight line embedding of G where the vertices are on a $2n-4 \times n-2$ grid [FPP90]. We now scale this graph by a polynomial factor large enough to ensure two properties. First, for every edge of G there is a portion of that edge which has length at least say 100 and the closest other edge or vertex is distance at least 100. Call this the *free zone* of the edge. Second, for each vertex of G, there is a ball centered at that vertex, such that this ball only intersects the at most 3 adjacent edges of that vertex, does not intersect the free zones of those edges, and the intersection points of the edges with the boundary of the ball are at least distance 10 apart from one another. Call this the *free zone* of the vertex. Note ensuring these two properties only requires scaling by a polynomial factor since the graph was initially embedded on a roughly $n \times n$ grid.

We now describe how to replace each edge of G with a sequence of unit squares. Roughly speaking this sequence of unit squares will be the unit grid cells that the edge overlaps, i.e. the standard pixelized representation of the edge. However, there will be several key differences, particularly inside the free zone of each edge and vertex, which we describe below. Outside the free zones we will simply include the unit squares of the grid cells intersected by the edge, except when the edge intersects 3 of the 4 grid cells adjacent to a grid point. In this case will will only include the diagonally adjacent squares. See Figure 3.1. Note in general there may be

Figure 3.1: Left: A portion of an edge. Middle: Grid cells intersected. Right: When three cells adjacent to a grid point are intersected, only the diagonally adjacent pair is kept.

Figure 3.2: Left: Two cases with consecutive grid points where 3 adjacent grid cells are intersected. Right: Iteratively removing intersected grid cells so that no grid point is adjacent to more than 2.

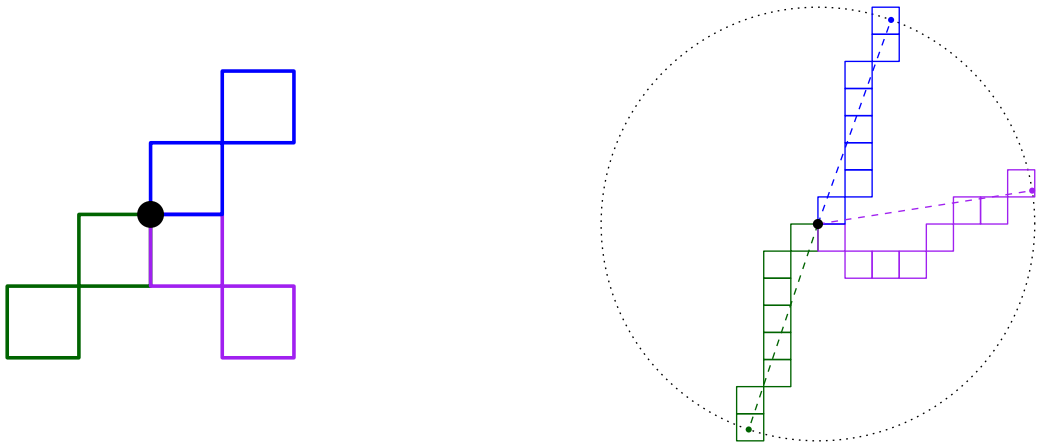

Figure 3.3: Left: The first two squares of each of the three edges adjacent to a vertex. Right: A routing of the square sequence for each adjacent edge which leads to the intersection point of that edge with the ball.

multiple grid points in a row where 3 of the 4 adjacent grid cell squares are intersected. Thus to be more precise, going from left to right, we iteratively remove the third adjacent grid cell square (again the square that is not in the diagonal pair), until all grid points are adjacent to at most 2 remaining intersected squares. See Figure 3.2.

Now we describe the construction within the free zone of a given vertex v, with adjacent edges e_1, e_2, e_3. (If v has fewer adjacent edges the construction is only simpler.) Within the free zone of v we cannot simply pixelize the edges as described above, since the angle of the adjacent edges might be such that say e_1 and e_2 initially pass through the same neighboring grid cell of v. Instead we enforce that the square sequence for each edge start on a distinct square adjacent to v, and moreover, the second square in the sequence for each edge continue in this diagonal direction from v. (This condition on the second square in the sequence of the edge ensures that squares from different edges are only adjacent at v itself.) See Figure 3.3. As described above when defining the free zone of v there exists a ball centered at v such that the points of intersection of e_1, e_2, e_3 with the boundary of this ball are at least distance 10 apart from one another. This ample spacing ensures that we can route the sequences of squares we are constructing for each edge such that squares from different edges stay at least distance 1 apart from each other (except at v itself) and that the square sequence for each edge ends up on its respective intersection point on the ball boundary. See Figure 3.3.

Finally, the last part of the construction concerns the free zone of each edge. So consider a given edge e. If e consists of a sequence of an odd number of squares after applying the above pixelization process to e along with the above modifications in the vertex free zones of its endpoints, then we leave the free zone of e untouched (i.e. it is just pixelized like the rest of e). However, if e consists of a sequence of an even number of squares then in the free zone we make the following modification so that the total number of squares is odd. We consider two cases. First, if within the free zone the sequence of squares has at least 6 consecutive squares which are horizontally adjacent (or 6 which are vertically adjacent), then we replace these 6 squares with the parity shifting gadget show in Figure 3.4 (where if the 6 squares where vertically adjacent we rotate the gadget 90 degrees).

Figure 3.4: Left: 6 horizontally adjacent squares. Right: Parity gadget replacing the 6 squares with 7 squares. The horizontal gaps between squares on the top and bottom rows have length exactly 1/2.

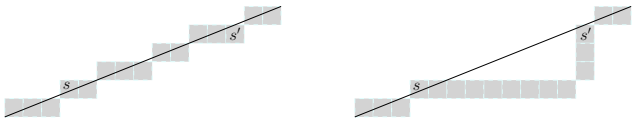

Figure 3.5: Left: Pixelized edge. Right: Replacing the portion of the pixelized edge between s and s' with an L shaped sequence of squares. Note the figure is not to scale, as the portion replaced did not have at least 12 squares.

Otherwise, if there are not 6 horizontally adjacent squares then we replace a portion of the square sequence in the free zone with an L shaped sequence, see Figure 3.5. In particular, viewing the squares in the sequence as ordered from left to right, pick a square s whose previous adjacent square is diagonally adjacent. Next pick a square s' which is at least 12 squares after

s in the sequence and such that the square after s' is diagonally adjacent. Now replace all squares between s and s' with an L shaped sequence consisting of single run of horizontally adjacent squares followed by a run of vertically adjacent squares (again see Figure 3.5). Now if this new sequence of squares between s and s' has a different parity than the original sequence between s and s' then we are done. Otherwise, either the horizontal or vertical portion of this L shaped sequence must have at least 6 squares and thus we can insert the same parity gadget described above into this portion of the L shaped sequence.

Let $0 < \varepsilon \ll 1/4$ be some value. For the final step in our construction, we now shrink all of the above created squares (about their respective centerpoints) such that two squares diagonally adjacent to the same grid point are distance 2ε apart from one another. (Note this means horizontally or vertically adjacent squares are $2\varepsilon/\sqrt{2}$ apart.)

So given an instance G, k of P3VC, we construct an instance $\mathcal{S}, \kappa$ of Problem 1 where $\mathcal{S}$ is determined from G as described above and $\kappa = k + (|\mathcal{S}| - |E|)/2$. We first argue if G has a vertex cover of size k then for our instance of Problem 1 there is a solution of radius ε. First, for any vertex v in the vertex cover we create a center, and place it at the location of v in the embedding. By the way we shrunk the squares, $B(v, \varepsilon)$ will intersect the (at most) three adjacent initial squares of v's adjacent edge sequences. We now cover the remaining squares with $(|\mathcal{S}| - |E|)/2$ centers. For any edge $e \in E$ let n_e be the number of squares used for e in the above construction. Observe that as we already placed centers at vertices corresponding to a vertex cover of the edges, at least one square of each edge is already covered, and so there are at most $n_e - 1$ consecutive squares that need to be covered. (Note $n_e - 1$ is even.) However, as consecutive squares are at most 2ε apart on each edge, these $n_e - 1$ squares can be covered with $(n_e - 1)/2$ balls of radius ε by covering the squares in pairs. Thus the total number of centers used is $k + \sum_{e \in E}(n_e - 1)/2 = k + (|\mathcal{S}| - |E|)/2 = \kappa$.

Now suppose the minimum vertex cover of G requires $> k$ vertices. In this case we argue that our instance of Problem 1 requires more than κ centers if we limit to balls with radius $< 1/4$. Call any two squares in $\mathcal{S}$ neighboring if they are consecutive on an edge or if they are squares on the v end of two edges adjacent to a vertex v. By construction, neighboring squares have distance $\leq 2\varepsilon$ from each other. For a pair of squares which are not neighboring their distance is at least $1/2$. Specifically, within the free zone of a vertex we ensured squares from different edges were at least unit distance apart (except at the vertex). Also, squares from differing edges remain at least unit distance apart outside of the free zones of vertices. For two squares from the same edge, the pixelization process enforces at least unit distance for non-adjacent squares. The same holds for inserting L shaped sequences in the free zone of an edge. Thus all that remains is the parity gadget, where the closest two non-adjacent squares can be is exactly $1/2$.

By the above, limiting to radius $< 1/4$ therefore implies that, other than at the (up to) three neighboring squares at a vertex, any ball either covers just a single square, or a pair of neighboring squares. An edge e with n_e squares thus requires at least $\lceil n_e/2 \rceil = 1 + (n_e - 1)/2$ balls to cover it. Moreover, a ball can only cover both a square of e and e' if those squares are on the v end of two edges adjacent to v. Let E_z be the subset of edges with at least one square covered by such a ball (i.e. a ball corresponding to a vertex), and let z be the number of such balls. Then the total number of balls required is

$$\geq z + \sum_{e \in E_z} (n_e - 1)/2 + \sum_{e \in E \setminus E_z} (1 + (n_e - 1)/2)$$
$$= z + (|\mathcal{S}| - |E|)/2 + |E \setminus E_z| = z + (\kappa - k) + |E \setminus E_z|,$$

which is more than κ when $z + |E \setminus E_z| > k$. Notice, however, there is a vertex cover of G of size $z + |E \setminus E_z|$, consisting of the vertices that z counted, and one vertex from either end of each edge in $E \setminus E_z$. Thus as the minimum vertex cover has size $> k$, we have $z + |E \setminus E_z| > k$ as desired.

Therefore, if we could approximate the minimum radius of our Problem 1 instance within any factor less than $\frac{1/4}{\varepsilon} = \frac{1}{4\varepsilon}$ then we can determine whether the corresponding vertex cover instance had a solution with $\leq k$ vertices. However, we are free to make $\varepsilon > 0$ as small we want, and thus $\frac{1}{4\varepsilon}$ as large as we want, so long as this quantity (or more precisely a lower bound on it) is computable in polynomial time. Thus we have the following theorem.

Theorem 2 *Problem 1 cannot be approximated within any factor in polynomial time unless* P = NP*, even when restricting to the set of instances in which $\mathcal{S}$ is a set of axis aligned squares of the same size.*

References

[Cha03] T. M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *J. Algorithms*, 46(2):178–189, 2003.

[FG88] T. Feder and D. H. Greene. Optimal algorithms for approximate clustering. In *20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 434–444. ACM, 1988.

[FPP90] H. De Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Comb.*, 10(1):41–51, 1990.

[GJ77] M. R. Garey and D. S. Johnson. The rectilinear steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.

[Gon85] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[HKR21] H. Huang, G. Klimenko, and B. Raichel. Clustering with neighborhoods. In *32nd International Symposium on Algorithms and Computation (ISAAC)*, volume 212 of *LIPIcs*, pages 6:1–6:17, 2021.

[HS85] D. S. Hochbaum and D. B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

[Kli23] G. Klimenko. *Convex hull simplification and geometric hardness*. The University of Texas at Dallas, 2023.

[XX10] G. Xu and J. Xu. Efficient approximation algorithms for clustering point-sets. *Computational Geometry*, 43(1):59–66, 2010.