Bounds for k-centers of point sets under L_∞-bottleneck distance

Mats Bierwirth
Dept. of Computer Science, ETH Zürich, Switzerland

Julia Hütte
Dept. of Computer Science, ETH Zürich, Switzerland

Patrick Schnider
Dept. of Mathematics and Computer Science, University of Basel and Dept. of Computer Science, ETH Zürich, Switzerland

Bettina Speckmann
Dept. of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Abstract

We consider the k-center problem on the space of fixed-size point sets in the plane under the L_∞-bottleneck distance. While this problem is motivated by persistence diagrams in topological data analysis, we illustrate it as a *Restaurant Supply Problem*: given n restaurant chains of m stores each, we want to place supermarket chains, also of m stores each, such that each restaurant chain can select one supermarket chain to supply all its stores, ensuring that each store is matched to a nearby supermarket. How many supermarket chains are required to supply all restaurants? We address this questions under the constraint that any two restaurant chains are so close under the L_∞-distance that they can be supplied by a single supermarket chain. We provide both upper and lower bounds for this problem and investigate its computational complexity.

Keywords and phrases k-center, Bottleneck Matching, Piercing Problems, Computational Geometry

Digital Object Identifier 10.57717/cgt.v5i3.91

Related Version arXiv:2503.02715

Funding *Mats Bierwirth*: Supported by the German Academic Scholarship Foundation
Julia Hütte: Supported by the German Academic Scholarship Foundation, Cusanuswerk and DAAD

Acknowledgements This work was initiated during the 21st Gremo's Workshop on Open Problems in Pura, Switzerland. The authors would like to thank the other participants for the lively discussions. The authors would further like to thank the reviewers for their helpful feedback, in particular the anonymous reviewer who improved the runtime of our algorithm in Theorem 7.

1 Introduction

The *k-center problem* is a classical problem in computational geometry [1, 6, 11, 12, 19]. Given a set P of points in some metric space X, the goal is to partition them into k parts $P_1, \dots, P_k$ such that for each part P_i there is some point $x_i \in X$, not necessarily in P, that is close to each point $p \in P_i$. The case $k = 1$ is of particular interest, as it relates to Helly's theorem: for $X = \mathbb{R}^d$, as balls in Euclidean space are convex, if for any $d+1$ points in P there is a point $x \in \mathbb{R}^d$ that has distance $\leq \delta$ to all of them, then there is a point $y \in \mathbb{R}^d$ that has distance $\leq \delta$ to all points in P.

While this problem has mainly been studied for points in Euclidean space, motivated by applications in *topological data analysis*, we are interested in the space of *persistence diagrams* under the *bottleneck distance*. We will give a quick introduction of both concepts and illustrate them in Figure 1; the familiar reader may freely skip to the next paragraph. A persistence diagram $D_{\mathcal{F}}$ consists of a set of (finitely many) off-diagonal tuples $T \subset \{(b, d) \in$

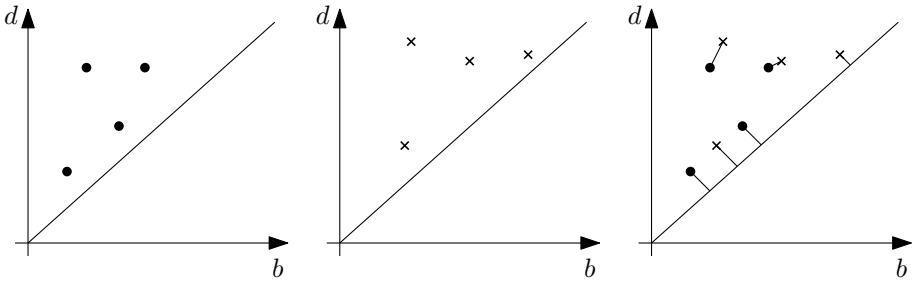

Figure 1 Two persistence diagrams and a bottleneck matching between them.

$(\mathbb{R}_{\geq 0} \cup \{\infty\})^2 : b < d\}$, where each tuple $(b, d) \in T$ represents the birth and death time of some topological feature under a specified filtration $\mathcal{F}$ of the underlying topological space. In addition, $D_{\mathcal{F}}$ also contains the diagonal, i.e. all tuples (b, d) where $d = b$. These do not represent specific topological features but allow for comparing any two persistence diagrams, even those with different numbers of off-diagonal tuples. The bottleneck distance is one way of imbuing the space of persistence diagrams with a metric. Let D_1, D_2 be two persistence diagrams. The bottleneck distance of D_1 and D_2 is defined as

$$d_b(D_1, D_2) := \inf_{\pi \in \Pi} \sup_{(b,d) \in D_1} \|(b, d) - \pi(b, d)\|_\infty,$$

where Π is the set of all bijections between D_1 and D_2. In other words, the bottleneck distance is the result of a min-max-optimization: each bijection π between D_1 and D_2 can be viewed as a matching between the off-diagonal tuples, where it is also allowed to match tuples to the diagonal. The cost is the length of the longest edge in the matching (in L_∞-distance). A bottleneck matching is a matching minimizing the cost, and the bottleneck distance is the cost of a bottleneck matching. Computing bottleneck distances has been studied extensively [2, 10, 13, 14]. For a more in depth explanation of these concepts, we refer the interested reader to textbooks on topological data analysis [4, 8, 9].

From a perspective of topological data analysis, a k-center for persistence diagrams would give a way to summarize a potentially large family of persistence diagrams by a much smaller summarizing family, a question that recently gained attention [3, 5, 15, 16, 18, 20]. Of course, if all the original persistence diagrams pairwise have large distance, then no reasonable summary is possible, so we will need some assumption of closeness. This is often done through a Helly-type condition, stating that for any h original objects there is a new one that is close to all of them. We can now formalize our question as follows: are there universal constants h and k such that if for any h persistence diagrams in a family P there is a persistence diagram which has bottleneck distance at most δ to all of them, then there are k persistence diagrams such that every persistence diagram in P has bottleneck distance at most δ to one of them?

There are two extremal cases of this question. On the one hand we can ask whether there is a value of h for which we can guarantee $k = 1$. On the other hand, we can ask whether $h = 2$ always suffices to bound k. As we will see in Section 2, the answer to the first question is "no" already for persistence diagrams with four off-diagonal points each. Most of this work thus focuses on the second question.

In the remainder of this paper we study a simplified version of the problem: instead of persistence diagrams, we only argue about finite point sets in $\mathbb{R}^2$ with a fixed number of points. This is a proper simplification since each instance on point sets can be transformed to an instance on persistence diagrams by considering those persistence diagrams that realize

the point sets as their off-diagonal points, offset from the diagonal by a consistent, sufficiently large margin, such that all pairwise bottleneck matchings never match any point to the diagonal. Then, solving the problem on the instance of persistence diagrams also gives a solution for the instance on the point sets. Thus, negative results for the instances on point sets transfer to the instances on persistence diagrams, but positive results do not.

To get a better intuition, we reframe the problem in terms of supplying restaurant chains with supermarket chains, which we call the *Restaurant Supply Problem*: In a city, there are n different restaurant chains that have m stores each. Their supply is secured by k supermarket chains that also have m stores each. All restaurants that belong to the same chain get their supply from the same supermarket chain. However, each restaurant within one chain gets it from a different store. This means that as soon as a supermarket chain gets chosen by a restaurant chain, each store gets matched to one specific restaurant that it supplies. We are interested in the number of supermarket chains needed to satisfy all restaurant chains. Formally, we define the following:

▶ **Definition 1.** *Let $R_1 = \{r_1^1, r_1^2, \dots, r_1^m\}, \dots, R_n = \{r_n^1, \dots r_n^m\}$ be n restaurant chains with m stores each in some metric space X. We say that a supermarket chain consisting of m supermarkets $s_1, \dots, s_m$ δ-*satisfies *(or equivalently just* satisfies*) a set of restaurant chains $R_a, \dots, R_b$ if there exists a relabeling of the stores in each R_i such that the distance between s_j and r_i^j is at most δ for each i and j. More generally, we say that k supermarket chains δ-satisfy the n restaurant chains $R_1, \dots, R_n$ if $R_1, \dots, R_n$ can be partitioned into k subfamilies, each of which can be δ-satisfied by a single supermarket chain.*

Of course, as was the case for the original problem on persistence diagrams, if the restaurant chains are operating sufficiently far from each other, then any supermarket chain can only satisfy one restaurant chain, so in general there are instances where we need n supermarket chains. For this reason, we focus on the case where the restaurant chains are in actual competition and operate close to each other. More specifically, we will assume that any h restaurant chains can be satisfied by a single supermarket chain. This assumption can make a big difference: consider the case where each restaurant chain only has a single store, and assume that distances are measured in Euclidean distance. Then a single supermarket satisfies some h restaurants whenever it lies in the intersection of the disks with radius δ centered at the locations of the restaurants. As these disks are convex, it follows by Helly's theorem that if any three of them have a common intersection, then all of them do. In other words, if any three restaurants can be satisfied by a single supermarket, then all restaurants can be satisfied by a single supermarket.

The Restaurant Supply Problem is now the optimization problem of minimizing the number of supermarket chains under this assumption. If we restrict ourselves to the L_∞-distance, then the disks of radius δ are axis-aligned squares and from Helly's theorem for boxes we get an even stronger statement for chains with only one store: if any two restaurants can be satisfied by a single supermarket, then all of them can. For this reason, we set $h = 2$ for most of this manuscript, that is, we assume that any two restaurant chains can be satisfied by a single supermarket chain.

2 No Helly-type theorem

In this section we show that in general, there is no intersection property on fixed sized subsets of the restaurant chains that imply that a single supermarket chain suffices.

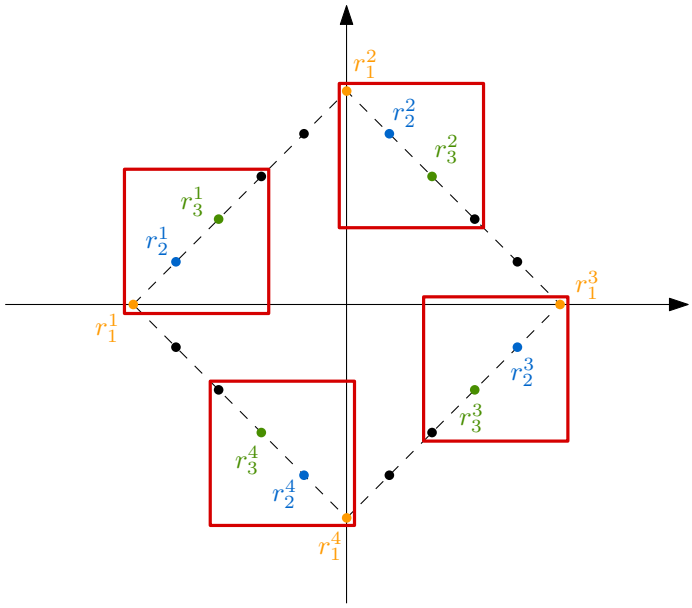

Figure 2 A placement of five restaurant chains such that any four can be satisfied by a single supermarket chain, but not all five simultaneously. A supermarket chain satisfying four of the restaurant chains is depicted in red.

▶ **Theorem 2.** *For any $h \geq 3$ there exists a placement of $n = h$ restaurant chains $\{r_1^1, \ldots, r_1^4\}, \ldots, \{r_n^1, \ldots, r_n^4\}$, with 4 stores each, such that any $h-1$ can be satisfied by a single supermarket chain, but satisfying all n chains requires at least two supermarket chains.*

Proof. We give concrete coordinates for the $4h$ restaurant locations. For an illustration, see Figure 2. Place the stores for the chain $\{r_i^1, \ldots, r_i^4\}$ at the locations $\{(-(h+1)+i, i-1), (i-1, (h+1)-i), ((h+1)-i, -i+1), (-i+1, -(h+1)+i)\}$. Set $\delta := (h-2)/2$, that is, any subfamily of the restaurant chains can be satisfied by a single supermarket if we can find four axis-parallel squares of side length $h-2$ such that for each restaurant chain, each store can be assigned to one of the squares containing it. In other words, all the stores assigned to a single square have pairwise L_∞-distance at most $h-2$.

We first claim that any $h-1$ of the restaurant chains can be satisfied by a single supermarket. Assume that we want to satisfy all restaurant chains except $\{r_i^1, \ldots, r_i^4\}$. For this, we note that $d_\infty(r_a^2, r_b^2) = |b-a|$ and $d_\infty(r_a^1, r_b^2) = |h-a+b|$. Thus, the restaurants $\{r_{i+1}^1, r_{i+2}^1, \ldots, r_h^1, r_1^2, \ldots, r_{i-1}^2\}$ have pairwise distance at most $h-2$. In particular, they can be assigned to a single square, which then contains exactly one store per relevant restaurant chain. The other three squares can be placed symmetrically.

It remains to show that not all restaurant chains can be satisfied by a single supermarket chain. For this, we first note that $d_\infty(r_{i+1}^1, r_i^2) = h-1$, $d_\infty(r_{i+1}^1, r_i^3) = \max\{2h+1-2i, 2i-1\} \geq h$ and $d_\infty(r_{i+1}^1, r_i^4) = h$. In particular, for every i, r_i^1 must be assigned to the same supermarket store as r_{i+1}^1. But then, all of $\{r_1^1, \ldots, r_{h+1}^1\}$ must be assigned to the same supermarket store, which is not possible as $d_\infty(r_1^1, r_{h+1}^1) = h-1$. ◀

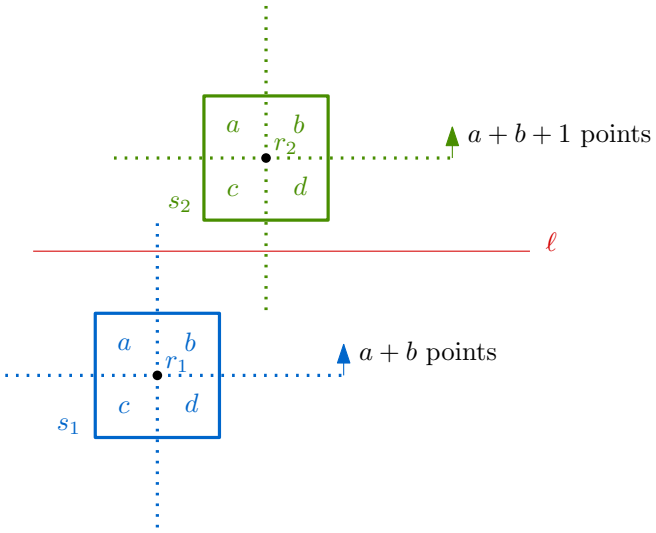

Figure 3 Two restaurants annotated with the same index from chains in the same permutation split the plane into quadrants with the same number of restaurants.

3 An upper bound

We now give an upper bound on the number of supermarkets that are sufficient to satisfy all restaurant chains.

▶ **Theorem 3.** *Let $\{r_1^1, \dots, r_1^m\}, \dots, \{r_n^1, \dots, r_n^m\}$ be n restaurant chains with m stores each, such that any two of them can be satisfied by a single supermarket chain. Then all of them can be satisfied by $k = m!$ supermarket chains.*

Before proving the theorem, we discuss a few helpful observations. For each restaurant chain, we consider its m stores and annotate them with the numbers $1, \dots, m$ from left to right. Now consider the permutation π obtained when enumerating the points of that restaurant chain from bottom to top. If any two points have the same x- or y-coordinate, we break ties lexicographically. We claim that all restaurant chains from the same permutation can be satisfied by just one supermarket chain.

▶ **Lemma 4.** *For two restaurants r_1, r_2 annotated with the same index from chains R_1, R_2 in the same permutation, their distance is at most 2δ.*

Proof. The restaurants r_1, r_2 split the plane into four quadrants each, see Figure 3. Since both stores are annotated with the same index and belong to chains from the same permutation, their quadrants contain the same number of points. Let a be the number of points in the top-left, b in the top-right, c in the bottom-left and d in the bottom-right quadrant. Furthermore, let s_1 and s_2 be the δ-balls (which are squares in L_∞) around r_1 and r_2. For sake of contradiction, assume that they do not intersect. Then there exists either a horizontal or vertical separation line between them. Without loss of generality consider the case of a horizontal separation line ℓ and let r_2 be above r_1.

By the Helly-like assumption of our theorem, R_1, R_2 can be satisfied using only one supermarket chain. Thus, their stores can be matched in a way such that the distance of the longest matching edge is at most 2δ. There are at least $a+b+1$ restaurants in R_2 (the upper two quadrants plus r_2 itself) that can in such a matching only be matched to restaurants

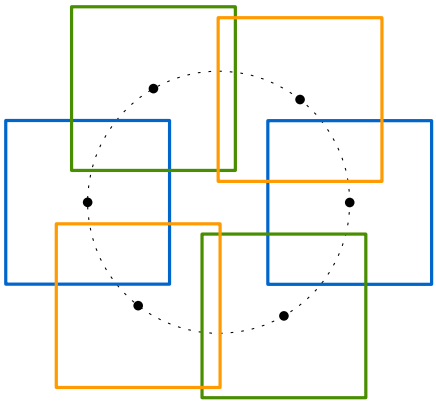

Figure 4 A city with three restaurant chains, each consisting of two stores placed on antipodal points.

strictly above ℓ. However, R_1 can contain at most $a + b$ such restaurants. By contradiction, s_1 and s_2 intersect. ◀

Proof of Theorem 3. By Lemma 4 any two stores annotated with the same index from chains in the same permutation are at most distance 2δ apart. In other words, their δ-balls intersect, and thus, from Helly's theorem for boxes, all restaurant chains with the same index can be satisfied by a single supermarket chain. Thus, by assigning one supermarket chain per permutation, the result follows. ◀

4 A lower bound

Unfortunately, the bound proven in the previous section has a superexponential dependence on the number of stores per chain m. As it turns out, no subexponential bound exists.

▶ **Theorem 5.** *Let n and m be fixed but arbitrary. Then there exists a placement of n restaurant chains $\{r_1^1, \ldots, r_1^m\}, \ldots, \{r_n^1, \ldots, r_n^m\}$, with m stores each, such that any two can be satisfied by a single supermarket chain, but satisfying all n chains requires $k \geq \min\{n, \lfloor e^{m/(27+\varepsilon)} \rfloor\}/2$ supermarket chains, for any fixed constant $\varepsilon > 0$.*

Proof. The proof is based on the following construction, an example where a single supermarket chain does not suffice. Consider six points arranged on a circle such that any two squares of radius δ centered at the points intersect if and only if the points are neighbors on the circle. These points will be locations of restaurants in our construction, and we refer to this arrangement as a city; an example is illustrated in Figure 4.

Since any two restaurant chains can be satisfied by a single supermarket chain, the minimum number of supermarket chains required to satisfy n restaurant chains is at most $n/2$. Thus, assume $n \geq e^{m/(27+\varepsilon)}$. Let $n_{\max} = \lfloor e^{m/(27+\varepsilon)} \rfloor$. We construct a placement for the first $n_{\max}$ restaurant chains that requires at least $n_{\max}/2$ supermarket chains to satisfy. This then entails the theorem.

Consider $\lfloor m/2 \rfloor$ cities, sufficiently far away from each other to not interfere with one another. In each city, each restaurant chain has to choose two of the six possible locations for its restaurants. We enforce that any restaurant chain chooses two antipodal points, leaving three options, among which each restaurant chain chooses one uniformly at random, independent of the choices of the other restaurant chains and also independent of its own choices in other cities.

We show that for any three restaurant chains, with high probability there exists a city where these three restaurant chains can not be satisfied by a single supermarket chain. This then entails that $n_{\max}/2$ supermarket chains are needed to satisfy all restaurant chains.

For a given city and three fixed restaurant chains there are 27 possible store placements of these three chains within this city. For six of those, namely the ones where all three antipodal pairs are chosen, one supermarket chain is not sufficient to satisfy the three restaurant chains (see Figure 4). Since each placement is equally likely, the probability of this happening for some fixed set of three restaurant chains is $2/9$. Using independence of the cities and the inequality $1 - x \leq e^{-x}$, the probability that all three chains can be satisfied in all cities is bounded by

$$\mathbb{P}[\text{Three restaurant chains satisfied}] = (1 - 2/9)^{m/2} \leq e^{-m/9} \leq \lfloor e^{m/9} \rfloor^{-1} = n_{\max}^{-3-\varepsilon'}$$

for some constant $\varepsilon' > 0$ dependent on ε. Applying a union bound over the at most $n_{\max}^3$ restaurant chain triples, the probability that some triple can be satisfied by a single supermarket chain is $n_{\max}^{-\varepsilon'}$, which is strictly less than one. Thus, with non-zero probability, this procedure produces a placement where no triple can be satisfied by a single supermarket chain. ◀

5 Computational complexity

In this section we analyze the computational complexity of deciding whether all restaurant chains can be satisfied by a single supermarket chain. In both our results it will be helpful to take a graph theoretic viewpoint on the problem: we define the *restaurant graph* G whose vertex set is the set of all restaurant stores, and where two stores r_a^b and r_x^y of different chains are connected whenever their δ-balls intersect. The vertex set of this graph partitions naturally into m-sets of vertices that correspond to the m stores of a restaurant chain. We interpret the different chains as colors and say that a clique C in G is a *colorful clique* if it contains exactly one vertex of each m-set associated to a restaurant chain.

▶ **Lemma 6.** *Let $\{r_1^1, \dots, r_1^m\}, \dots, \{r_n^1, \dots, r_n^m\}$ be n restaurant chains with m stores each. Then all of them can be satisfied by a single supermarket chain if and only if the vertex set of the restaurant graph G can be partitioned into m colorful cliques.*

Proof. Assume there is a placement of the m supermarket stores $s_1, \dots, s_m$ such that each restaurant r_j^i is satisfied by the store s_i. In particular, any two vertices r_a^i and r_b^i are connected and we can thus partition the vertex set of G into m colorful cliques. On the other hand if such a partition into colorful cliques exists, then for any two stores in this clique their δ-balls intersect. Thus, by Helly's theorem for boxes, all of them intersect, which means that we can place a supermarket store in this intersection to satisfy all the stores in the clique. ◀

The following can be proven using standard methods.

▶ **Theorem 7.** *Let $\{r_1^1, r_1^2\}, \dots, \{r_n^1, r_n^2\}$ be n restaurant chains, each having two stores, such that any two of them can be satisfied by a single supermarket chain. Then we can decide in time $O(n^2)$ whether all of them can be satisfied with a single supermarket chain.*

Proof. First, build the restaurant graph G. By Lemma 6, we want to decide whether the vertex set of G can be partitioned into two colorful cliques. Consider now the subgraph induced by the four stores of two restaurant chains A and B. This is a subgraph of the complete bipartite graph $K_{2,2}$. If it is not the complete bipartite graph $K_{2,2}$ then it contains

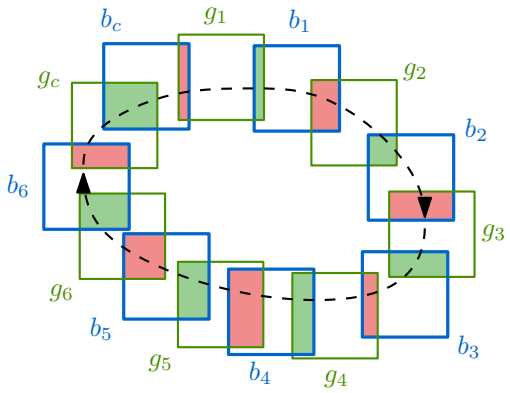

Figure 5 Placing supermarkets in the green (red) regions sets the variable to "true" ("false").

at most one perfect matching. In this case, we say that the pair (A, B) of vertex pairs is *forced*. If the subgraph does not contain any perfect matching, then we call the pair (A, B) *incompatible*. Note that as any two restaurant chains can be satisfied by a single supermarket chain, G does not contain any incompatible pairs (yet).

Given a forced pair (A, B) on the vertices $a_1, a_2 \in A$ and $b_1, b_2 \in B$, we perform the following *contraction*: assume without loss of generality that the unique perfect matching between A and B connects a_1 with b_1 and a_2 with b_2. Create a new vertex pair $C = (c_1, c_2)$ and remove A and B from the graph G. For each other vertex v in G, connect v to c_i if and only if v was connected to both a_i and b_i. Note that in this contraction step we might create an incompatible pair involving C.

Our algorithm now proceeds as follows: as long as there are forced pairs in G, contract them. As soon as there is an incompatible pair, return that there is no solution. If we end up in the situation that no pair is forced or incompatible, return that there is a solution.

This algorithm runs in time $O(n^2)$: building the graph takes time $O(n^2)$ and each contraction takes time $O(n)$. As in every contraction, we decrease the size of the vertex set by 2, we perform at most $O(n)$ many contractions.

It follows from the construction of the contraction step that the original graph can be partitioned into colorful cliques if and only if the contracted graph can, which proves the correctness of our algorithm. ◀

Let us mention here that we did not try to optimize the runtime and it is thus likely that it can still be improved. Further, it is an interesting problem whether for a larger constant number of stores we can still solve the problem in polynomial time.

▶ **Question 8.** *Given n restaurant chains, each having m stores, where m is a constant, such that any two of them can be satisfied by a single supermarket chain, is there a polynomial time algorithm to decide whether all of them can be satisfied by a single supermarket chain?*

On the other hand, if the number of stores is unbounded, then the problem becomes NP-complete, even if there are only three restaurant chains.

▶ **Theorem 9.** *Let $\{r_1^1, \ldots, r_1^m\}, \{r_2^1, \ldots, r_2^m\}, \{r_3^1, \ldots, r_3^m\}$ be 3 restaurant chains with m stores each, such that any two of them can be satisfied by a single supermarket chain. Then it is NP-complete to decide whether all of them can be satisfied with a single supermarket chain.*

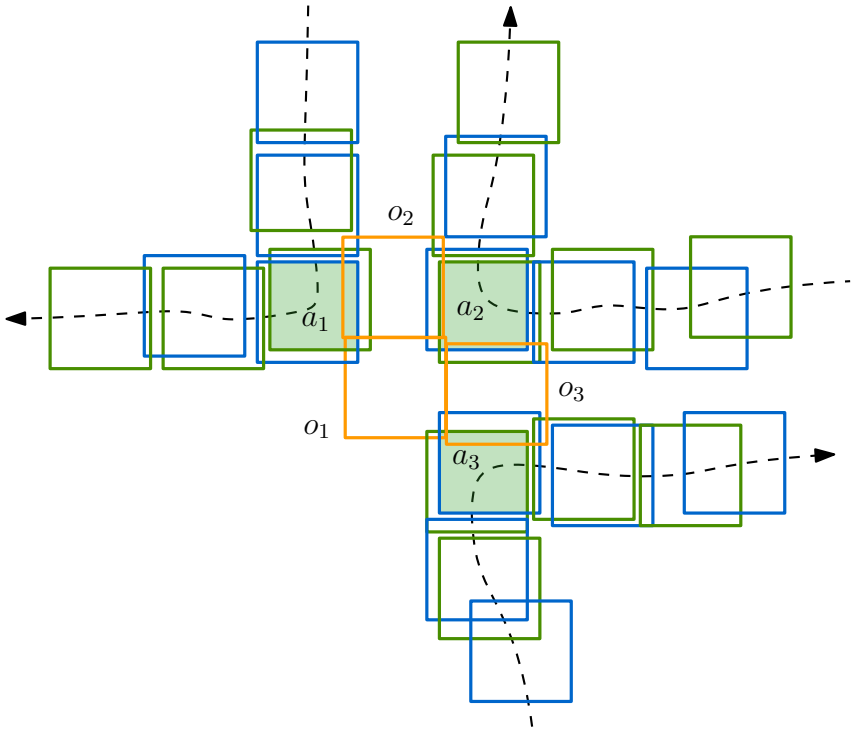

Figure 6 A clause gadget.

Proof. The containment in NP follows from Lemma 6. In order to show that the problem is NP-hard, we use a reduction from planar 3-SAT [17]. We describe our construction in terms of the δ-balls. For each of the three restaurant chains we give a different color, namely blue, green and orange. Thus, for our reduction, given a planar SAT formula φ with t clauses we want to place m blue squares, m green squares and m orange squares, where m is a number that is polynomial in t and will be determined later, such that we can partition them into colorful triples with common intersections if and only if φ is satisfiable. We now describe such a placement.

Variable gadgets. For an illustration of the variable gadget, see Figure 5. For each variable in the formula φ we use c green and blue squares $g_1, \dots, g_c$ and $b_1, \dots, b_c$, where c is some large enough number that is polynomial in t and will be determined later. We arrange the squares in a cycle $g_1, b_1, g_2, b_2 \dots, g_c, b_c$ infused with a clockwise direction such that g_i intersects exactly b_{i-1} and b_i and b_i intersects exactly g_i and g_{i+1} (we set $b_0 = b_c$ and $g_{c+1} = g_1$). The exact positioning of this cycle is determined later, when the gadgets are linked. Note that there are exactly two types of placements of c supermarkets to satisfy the involved restaurants: we either place them in the intersection $g_i \cap b_i$ or in the intersections $b_i \cap g_{i+1}$. We will interpret the former as setting the variable to "true" and the latter as setting it to "false".

Clause gadgets. For an illustration of the clause gadget, see Figure 6. Consider the clause (ℓ_1, ℓ_2, ℓ_3), where ℓ_i is a literal of the variable x_i, that is, it is either x_i or $\neg x_i$. Denote by a_i an intersection in the variable gadget of x_i corresponding to a placement of supermarkets setting ℓ_i to "true". We now place an orange square o_1, which we call the *main clause square*, in such a way that it intersects exactly all three a_i's, that is, it intersects the a_i's, but no other intersections of squares in the variable gadgets. Note that in this way, a supermarket

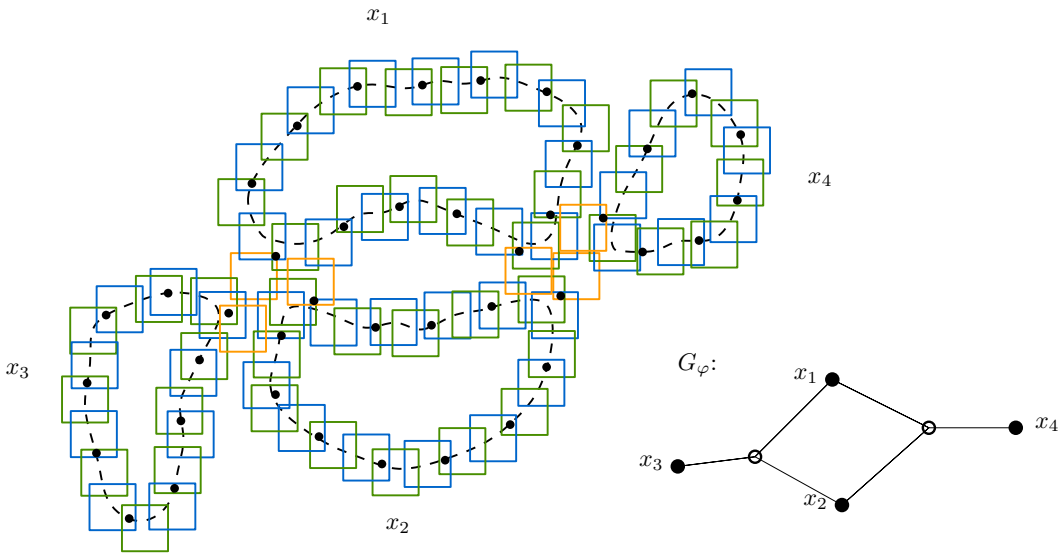

Figure 7 All gadgets for the (very small) formula $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4)$. The black points describe a placement of supermarkets corresponding to setting all variables to "true".

satisfying o_1 needs to lie in one of the intersections a_i, meaning that the corresponding literal is true. We place two more orange squares o_2, intersecting a_1 and a_2 as well as the intersections before a_1 on the variable gadgets of x_1 and the intersection after a_2 on the variable gadgets of x_2, and o_3, intersecting a_2 and a_3 as well as the intersections before a_2 on the variable gadgets of x_2 and the intersection after a_3 on the variable gadgets of x_3, see Figure 6. The three orange squares o_1, o_2 and o_3 will ensure that at least one of the relevant literals is true in any solution.

Linking the gadgets. For an illustration of this step, see Figure 7. Consider the variable-clause graph G_φ of the formula φ, i.e., the graph whose vertices are the variables and clauses of φ, with a connection between variable x_i and clause C_j if the literal ℓ_i appears in C_j. As φ is a planar 3-SAT formula, this graph is planar. Fix an embedding of G_φ on a polynomial grid, e.g. by the algorithm of de Fraysseix, Pach and Pollack [7]. For each vertex corresponding to a variable, route the variable gadget along its edges and place a clause gadget at each clause vertex. As the embedding is on a polynomial grid, this can be done with polynomially many green and blue squares. By construction, the number of green and blue squares is equal. Further, we have so far used 3 orange squares for each clause.

Adding the remaining orange squares. For an illustration of the placement of the additional orange squares, see Figure 8. So far we have only added three orange squares for each clause, that is, one per variable-clause incidence. We first add one more orange square for each such incidence as follows: let b_i be the blue square and g_j the green square of variable cycle x such that $b_i \cap g_j$ intersects the main clause square of the clause gadget. Note that depending on whether x or $\neg x$ appears in the clause either $j = i$ or $j = i-1$. Consider the two intersections before and after $b_i \cap g_j$ on the variable cycle. We place an orange square in such a way that it intersects all four of those intersections (see Figure 8). We further remember b_i and b_{i-1} as *marked*. Note that we have two marked blue squares and two orange squares per variable-clause incidence. Finally, we add an orange square at the location of each unmarked blue square. We thus have the same number of orange squares as blue squares and hence also the same number as green squares.

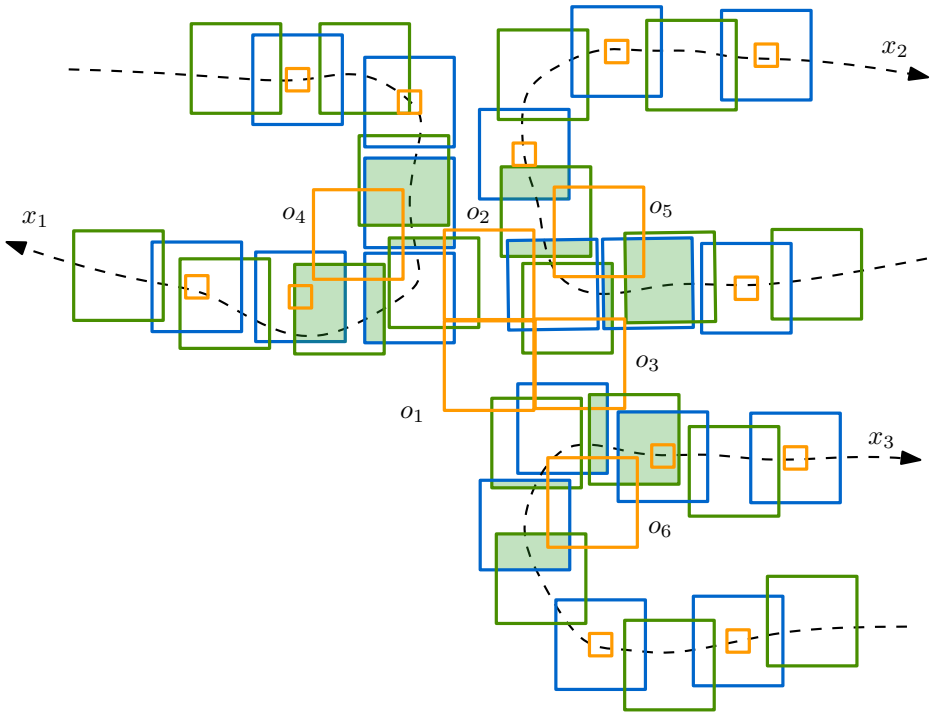

Figure 8 The placement of the remaining orange squares. The orange squares that are copies of blue squares are drawn small.

Correctness. We first show that we have constructed a valid instance of the problem, that is, that any two restaurant chains can be satisfied by a single restaurant chain. For the green and blue squares this follows from the construction: all the variable cycles are independent and each cycle can be satisfied in two ways. For the blue and orange squares, all except the marked squares are copies of each other, so it suffices to show that the marked squares can be satisfied. For each clause containing literals of the variable x_1, x_2 and x_3 label the six orange squares that are not copies $o_1, ..., o_6$, where o_1 is the main clause square, o_2 touches the variable cycles for x_1 and x_2, o_3 touches the variable cycles for x_2 and x_3 and o_{3+k} is the square added for the clause-variable incidence of variable x_k (See Figure 8). Then o_k intersects b_i^k and o_{3+k} intersects b_{i-1}^k, where b_i^k, b_{i-1}^k are the two marked blue squares for the variable clause incident of this clause and variable x_k. Thus it is possible to satisfy both the blue and the orange restaurant chains by a single supermarket chain.

A similar argument works for the green and orange squares. We match orange squares that are copies of blue squares b_j to g_j. Thus we again only need to satisfy that the green squares that have marked blue counterparts can be matched to the remaining orange squares. Consider again some clause gadget, and let g_j^k be the green square of variable cycle x_k that intersects the main clause square. By construction $j = i$ or $j = i - 1$, where b_i^k is the blue square of variable cycle x_k that intersects the main clause square. Both b_i^k and b_{i-1}^k are marked. Thus g_j^k is still unmatched. We match it to o_k. The last unmatched green square of variable cycle x_k is either g_{j-1}^k or g_{j+1}^k both are intersected by o_{3+k} and can thus also be matched. Thus it is possible to satisfy both the green and the orange restaurant chains by a single supermarket chain.

Now consider a valid assignment A of the formula φ. We need to show that this assignment induces a supermarket chain that satisfies all three restaurant chains. The general idea is to

place supermarkets in the true intersection of variable cycles assigned true by A and vice versa. This trivially satisfies all blue and green squares. We need to show that this can be done in such a way that each orange square is satisfied by a unique supermarket. We start by only placing supermarkets in intersections of unmarked blue squares, that is, those for which we placed orange copies. Clearly this partial placement satisfies all unmarked blue squares, all orange copies of blue squares. Which green squares are satisfied depends on the assignment A and the type of clause. For example, if blue squares b_i and b_{i-1} are marked, then either g_i and g_{i-1} or g_{i+1} and g_i are unsatisfied by the partial placement. All other green squares will be satisfied.

In total two squares per variable-clause incident per color remain unsatisfied by such a partial placement. Thus, six more supermarkets must be placed per clause in such a way that they satisfy the rest of the squares. Let c be some clause touching x_1, x_2 and x_3. Furthermore let b_i^1, b_{i-1}^1, b_j^2, b_{j-1}^2 and b_k^3, b_{k-1}^3 be the squares marked on the three variables cycles for this clause. Without loss of generality, assume x_1 provides the true literal to c. Then by construction, the intersection of o_1 with b_i^1 and either g_i^1 or g_{i+1}^1 is non empty and we can place a supermarket in this intersection to satisfy these three squares. Note that in case g_{i+1}^1 is part of this intersection, then g_{i+1}^1 has not been satisfied by the partial supermarket placement, since by assumption the supermarkets for variable cycle x_1 are placed in the $b_i^1 \cap g_{i+1}^1$ intersections. This leaves b_{i-1}^1 and either g_i^1 or g_{i-1}^1. By construction o_4 intersects both $b_{i-1}^1 \cap g_i^1$ and $b_{i-1}^1 \cap g_{i-1}^1$. Thus we can place a supermarket in the corresponding intersection of the two unsatisfied squares on the x_1 cycle with o_4.

We proceed in a similar vein with x_3. We place supermarkets in the intersection $o_3 \cap b_k^3 \cap g_{k+1}^3$ and $o_6 \cap b_{k-1}^3 \cap g_k^3$ if g_{k+1}^3 was left unsatisfied by the partial placement and $o_3 \cap b_k^3 \cap g_k^3$ and $o_6 \cap b_{k-1}^3 \cap g_{k-1}^3$ otherwise. Note that this is again possible by the construction of o_3 and o_6. Lastly, for x_2 the same argument as for x_3 works just using the remaining orange squares o_2 and o_5.

Finally, any placement of supermarket stores that satisfy all three restaurant chains can be mapped back to an assignment that satisfies φ, by setting all literals that are matched to the same supermarket as the main clause square of some clause to true and all others to false. As discussed any supermarket chain satisfying a variable cycle must either take all intersections $g_i \cap b_i$ or all intersections $b_i \cap g_{i+1}$ of that cycle. Thus no variable cycle may match main clause squares with both its literals. ◀

6 Conclusion

We investigated the k-center problem on the space of fixed-sized point sets in the plane under the L_∞-bottleneck distance, illustrated as the Restaurant Supply Problem. Our original question was whether a Helly-type theorem exists in this setting. We ruled this out by showing that even if any h restaurant chains can be satisfied by one supermarket chain, in general, exponentially many are needed to satisfy all of them. On the other hand, we showed an upper bound of $m!$ supermarket chains to cover all restaurant chains under the assumption that any two of them have small distance. From a computational perspective, we gave a polynomial-time algorithm for $m = 2$ stores per chain to determine whether a single supermarket chain suffices. However, for large m, deciding whether three restaurant chains can be satisfied by a single supermarket chain is NP-complete.

Both the non-existence of a Helly-type theorem and the NP-completeness result extend to persistence diagrams, which originally motivated our work. In contrast, our positive existence and algorithmic results do not directly carry over to persistence diagrams. An interesting

future direction is to ask whether meaningful k-center results can be obtained directly for persistence diagrams under the bottleneck distance, and if so, whether such centers can be approximated efficiently. Moreover, in the setting of this paper, tightening the gap between our upper and lower bounds remains an open question.

References

1 Mihai Bundefineddoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, page 250–257. Association for Computing Machinery, 2002. doi:10.1145/509907.509947.

2 Rainer E. Burkard and Ulrich Derigs. *The Bottleneck Matching Problem*, pages 60–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 1980. doi:10.1007/978-3-642-51576-7_5.

3 Yueqi Cao, Prudence Leung, and Anthea Monod. k-means clustering for persistent homology. *Advances in Data Analysis and Classification*, pages 1–25, 2024. doi:10.1007/s11634-023-00578-y.

4 Gunnar Carlsson and Mikael Vejdemo-Johansson. *Topological data analysis with applications*. Cambridge University Press, 2021. doi:10.1017/9781108975704.

5 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. In *Proceedings of the twenty-first annual Symposium on Computational Geometry (SoCG '05)*, pages 263–271, 2005. doi:10.1007/s00454-006-1276-5.

6 Graham Cormode and Andrew McGregor. Approximation algorithms for clustering uncertain data. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '08, page 191–200. Association for Computing Machinery, 2008. doi:10.1145/1376916.1376944.

7 Hubert De Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10:41–51, 1990. doi:10.1007/BF02122694.

8 Tamal Krishna Dey and Yusu Wang. *Computational topology for data analysis*. Cambridge University Press, 2022. doi:10.1017/9781009099950.

9 Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010. doi:10.1090/mbk/069.

10 Alon Efrat and Alon Itai. Improvements on bottleneck matching and related problems using geometry. In *Proceedings of the twelfth annual Symposium on Computational Geometry (SoCG '96')*, pages 301–310, 1996. doi:10.1145/237218.237399.

11 Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996. doi:10.5555/3001460.3001507.

12 Gabriel Y. Handler and Pitu B. Mirchandani. Location on networks: Theory and algorithms. *Networks*, 10(4):293–314, 1980. doi:10.1002/net.3230100408.

13 A. Karim Abu-Affash, Paz Carmi, Matthew J. Katz, and Yohai Trabelsi. Bottleneck non-crossing matching in the plane. *Computational Geometry*, 47(3, Part A):447–457, 2014. doi:10.1016/j.comgeo.2013.10.005.

14 Matthew J. Katz and Micha Sharir. Bottleneck matching in the plane. *Computational Geometry*, 112:101986, 2023. doi:10.1016/j.comgeo.2023.101986.

15 Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. Geometry Helps to Compare Persistence Diagrams. *ACM Journal of Experimental Algorithmics*, 22:1–20, 2017. doi:10.1145/3064175.

16 Théo Lacombe, Marco Cuturi, and Steve Oudot. Large scale computation of means and clusters for persistence diagrams using optimal transport. *Advances in Neural Information Processing Systems*, 31, 2018. doi:10.5555/3327546.3327645.

17 David Lichtenstein. Planar formulae and their uses. *SIAM journal on computing*, 11(2):329–343, 1982. doi:10.1137/0211025.

18 Andrew Marchese, Vasileios Maroulas, and Josh Mike. K-means clustering on the space of persistence diagrams. In *Wavelets and Sparsity XVII*, volume 10394, pages 218–227. SPIE, 2017. doi:10.1117/12.2273067.

19 Nimrod Megiddo and Kenneth J. Supowit. On the Complexity of Some Common Geometric Location Problems. *SIAM Journal on Computing*, 13(1):182–196, 1984. doi:10.1137/0213014.

20 Jules Vidal, Joseph Budin, and Julien Tierny. Progressive Wasserstein Barycenters of Persistence Diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):151–161, 2020. doi:10.1109/TVCG.2019.2934256.