Bichromatic 2-center of pairs of points *

Esther M. Arkin[1], José Miguel Díaz-Báñez[2], Ferran Hurtado[3], Piyush Kumar[4], Joseph S. B. Mitchell[1], Belén Palop[5], Pablo Pérez-Lantero[6], Maria Saumell[7], and Rodrigo I. Silveira[3,8]

[1] Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY, USA
{estie,jsbm}@ams.stonybrook.edu
[2] Departamento Matemática Aplicada II, Universidad de Sevilla, Spain
dbanez@us.es
[3] Dept. de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Spain
{ferran.hurtado,rodrigo.silveira}@upc.edu
[4] Dept. of Computer Science, Florida State University, Tallahassee, FL, USA
piyush@acm.org
[5] Departamento de Informática, Universidad de Valladolid, Spain
b.palop@infor.uva.es
[6] Esc. de Ingeniería Civil en Informática, Universidad de Valparaíso, Chile
pablo.perez@uv.cl
[7] Department of Mathematics and European Centre of Excellence NTIS (New Technologies for the Information Society), University of West Bohemia, Czech Republic
saumell@kma.zcu.cz
[8] Departamento de Matemática & CIDMA, Universidade de Aveiro, Portugal
rodrigo.silveira@ua.pt

Abstract. We study a class of geometric optimization problems closely related to the 2-center problem: Given a set S of n pairs of points in the plane, for every pair, we want to assign color red to a point of the pair and color blue to the other point in order to optimize the radii of the minimum enclosing ball of the red points and the minimum enclosing ball of the blue points. In particular, we consider the problems of minimizing the maximum and minimizing the sum of the two radii of the minimum enclosing balls. For each case, minmax and minsum, we consider distances measured in the L_2 and in the L_∞ metrics.

1 Introduction

In this paper we consider the following geometric optimization problem:

The 2-Center Color Assignment problem: Given a set S of n pairs of points in the plane, for each pair of S choose one point to be red and the other to be blue, in such a way that a function of the size of the minimum enclosing balls of the set of red points R and the set of blue points B is minimized.

We consider two optimization criteria: the first one is to minimize the maximum of the radii of the minimum enclosing balls of R and B, respectively, while the second one is to minimize their sum. For each criterion, we study the problem for both the L_∞ and the L_2 metrics. Thus, we consider four variants of the 2-Center Color Assignment problem that will be referred to as: the MinMax-L_∞ problem, the MinMax-L_2 problem, the MinSum-L_∞ problem, and the MinSum-L_2 problem.

A natural variant of these problems is the Pairs of Points 1-Center problem, in which the goal is to determine a minimum-radius ball that encloses at least one point of each of the pairs. We consider the corresponding versions of this problem in the L_∞ and L_2 metrics. We refer to them as the Pairs of Points L_∞ 1-Center problem and the Pairs of Points L_2 1-Center problem, respectively.

* A preliminary version of this paper appeared in Proc. 10th Latin American Symposium on Theoretical Informatics, 2012, pp. 25-36.

Motivation. In addition to being a natural variant of the fundamental 2-CENTER problem from facility location, our problem is motivated by a problem in "chromatic clustering", the CHROMATIC CONE CLUSTERING problem, which arises in certain applications in biology, as studied by Ding and Xu [19] and described in the related work section below. We also mention below connections between our model and certain problems in the study of imprecise (or "indecisive") points.

Another motivating view of our problem comes from a transportation problem in which there are origin/destination pairs of points between which traffic flows. We have the option to establish a special high-priority traffic corridor, modeled as a straight segment, which traffic flow is required to utilize in going between pairs of points. The corridor offers substantial benefit in terms of safety and speed. Our goal is to locate the corridor in such a way that we minimize off-corridor travel when traffic between origin/destination pairs utilizes the corridor. Models dealing with alternative transportation systems have been suggested in location theory [7], and simplified mathematical models have been widely studied in order to investigate basic geometric properties of urban transportation systems [2]. Recently, there has been an interest in facility location problems derived from urban modeling. In many cases one is interested in locating a highway that optimizes some given function that depends on the distance between elements of a given point set [5,11,18,28]. Specifically, in this work, we are motivated by an application in air traffic management, in which the use of "flow corridors" (or "tubes") has had particular interest. Flow corridors have been proposed as a potential means of addressing high demand routes by establishing dedicated portions of airspace designed for self-separating aircraft, requiring very little controller oversight [33,34,35,36]. Given a set S of *pairs* of points (origin/destination pairs) in the plane, we want to find two "centers", which define the endpoints of a corridor. Traffic travels from its origin to one endpoint of the corridor, follows the corridor to the other endpoint, then proceeds directly to its corresponding destination; see Fig. 1. Of course, the real air-traffic problem has to take into consideration many other issues, including traffic congestion, sector geometry, fuel consumption, flight dynamics, time, etc. Here, we consider a simplified model in which we assign each airport to one of the two endpoints of the flow corridor and must determine an optimal location for the flow corridor, with the objective of minimizing the distances from airports to their assigned endpoints. In this model, we assume that every pair must utilize the corridor, explicitly ruling out the possibility of going directly between the pair of points; such direct flights may result in unwanted traffic congestion outside the neighborhoods of the corridor endpoints.

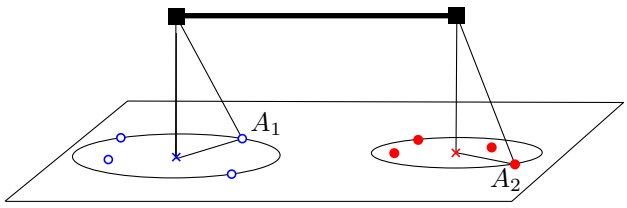


Fig. 1. Schematic of a flow corridor (bold segment) servicing air traffic between blue and red points (airports). The maximum distances between the airports of each color and their closest endpoint depend on the radii of the disks.

Related work. Both the 1-CENTER (also called MINIMUM ENCLOSING DISK) and the 2-CENTER problem have been widely studied for points in the plane. The 1-CENTER problem for n points in the plane is well known to be solvable in $O(n)$ time using techniques related to linear-programming and prune-and-search [15,21,31]. The 2-CENTER problem has received much attention in recent years; the current best known deterministic algorithm is due to Chan [14], and the current best randomized algorithm is due to Eppstein [22]. The RECTILINEAR 2-CENTER problem, in which the metric used is L_1 or L_∞, can be solved in linear time [20]. The discrete version was considered by Bespamyatnikh and Segal [9]. However, the restriction on the coloring of the pairs of points that

we have in this paper makes our problems rather different, and it seems that we cannot directly apply any similar methods to our case.

Our problem is also similar to the facility location problems with the objective of minimizing the maximum cost of the customers, where the cost of a customer is the minimum between the cost of using the facility and the cost of not using the facility [12]. However, the objective functions we use are more complex, leading to considerably more involved problems.

A problem closely related to ours is the Chromatic Cone Clustering problem of Ding and Xu [19]: Given a point set $\mathcal{G} = G_1 \cup G_2 \cup \ldots \cup G_n \subset \mathbb{R}^d$ formed by the n point sets $G_1, G_2, \ldots, G_n$, such that each G_j consists of k points of positive coordinates (i.e., points in the positive orthant), find k cones $C_1, C_2, \ldots, C_k$ with apex at the origin such that each C_i contains a distinct point from every point set G_j and the total amplitude of the cones is minimized. The authors give a $(1+\varepsilon)$-approximation algorithm for this problem by projecting the points of $\mathcal{G}$ into the unit sphere and finding spheres $C'_1, C'_2, \ldots, C'_k$ of minimum total radius such that each contains a distinct point from every G_j. Observe that for $k = 2$ the input is precisely a set of pairs of points, and each of the two output spheres (defining the cones) contains a different point from each pair. Then the Chromatic Cone Clustering problem for $k = 2$ is directly related to the MinSum-L_2 problem. The running time of Ding and Xu's algorithm, stated for high dimensions d, has the parameter $1/\varepsilon$ as an exponent. In comparison, our problem, formulated in the two-dimensional plane, is solved deterministically and the approximation algorithms have running times polynomial in both n and $1/\varepsilon$.

The problems studied in this paper can also be seen from the perspective of data imprecision. Data imprecision in computational geometry has received a lot of attention lately (see [30] and references therein). Even though most data imprecision models represent an imprecise point as a continuous region (e.g. a disk or rectangle), in the so-called *indecisive point* model [27], an imprecise point is given by a set of possible locations for the point. The setting studied here can be seen as dealing with indecisive points, having each of them exactly two possible locations. In fact, the minimum enclosing disk problem is one of the measures studied in [27], but their goal is not to compute the smallest possible disk (that would be closely related to our Pairs of Points 1-Center problem), but to compute the whole distribution of possible values for the size of the disk. Therefore their algorithms address a rather different problem than ours.

In the time since the first version of this paper appeared, there has been further related work studying the setting considered in this paper, that of problems on pairs of points. The paper by Díaz-Báñez et al. [17] studied computing the convex hull of a set of n pairs of points, where the goal is to find the smallest (or largest) convex polygon that contains at least one point from each pair. The same problem, among others, was studied in a more general setting in a paper by Consuegra et al. [16], where a class of problems called *avatar problems* are proposed. These are problems in which the input is a set of objects each having k copies (avatars), and one wants to compute some structure using (at least) one copy of each object. The case in which the objects are points in the plane and $k = 2$ corresponds to our setting. Consuegra et al. [16] study this setting for several geometric problems, namely segment intersection, convex hull, and minimum spanning trees.

Results. We present exact algorithms for all four variants of the 2-Center Color Assignment problem. In addition, we present a $(1+\varepsilon)$-approximation (with two variants that give different dependencies between n and ε) that works for both the MinMax-L_2 problem and the MinSum-L_2 problem, which gives simple and fast alternatives to the slower exact algorithms. We also solve the Pairs of Points L_∞ 1-Center problem and the Pairs of Points L_2 1-Center problem. The algorithms we propose for the latter one can be used to solve the MinSum problems. The running times of the algorithms are summarized in Table 1.

Notation. Set S denotes the set of n pairs of points. By C_R and C_B we denote the two balls that form an optimal solution, ball C_R covers the points colored red and ball C_B covers the points colored blue. Given a point u, we denote by $x(u)$ and $y(u)$ the x- and y-coordinates of u, respectively.

	MinMax	MinSum	Pairs of Points 1-Center
L_∞	$O(n)$	$O(n \log^2 n)$ worst case $O(n \log n)$ expected $\Omega(n \log n)$ lower bound	$O(n \log^2 n)$ worst case $O(n \log n)$ expected $\Omega(n \log n)$ lower bound
L_2	$O(n^3 \log^2 n)$ $O((n/\varepsilon^2) \log n \log(1/\varepsilon))$ $(1+\varepsilon)$-approx $O(n + (1/\varepsilon^6) \log^2(1/\varepsilon))$ $(1+\varepsilon)$-approx	$O(n^4 \log^2 n)$ $O((n/\varepsilon^5) \log(1/\varepsilon))$ $(1+\varepsilon)$-approx $O(n + (1/\varepsilon^7) \log^2(1/\varepsilon))$ $(1+\varepsilon)$-approx	$O(n^2 \log n)$

Table 1. Summary of the running times of the algorithms for the different variants of the problem.

Outline. The MinMax-L_∞ problem and the MinMax-L_2 problem are studied in Sections 2 and 3, respectively. In Section 4 we consider both the Pairs of Points L_∞ 1-Center problem and the Pairs of Points L_2 1-Center problem. In Sections 5 and 6 the MinSum-L_∞ problem and the MinSum-L_2 problem are solved, respectively. Finally, in Section 7, we point to future directions of research.

2 The MinMax-L_∞ problem

Let H denote the smallest axis-aligned rectangle covering S. Using the local optimality, we can assume, without loss of generality, that disks (squares) C_R and C_B have equal radius and that each of them has one of its vertices coinciding with a corner of H. We consider two fixed vertices of H and assume that C_R and C_B are anchored to them, respectively. For each pair (p, p') of S, let $r_{p,p'}$ be the smallest radius that C_R and C_B must have in order to satisfy that one element of (p, p') belongs C_R and the other element belongs to C_B. Observe that $r_{p,p'}$ can be computed in constant time (see Fig. 2). Therefore, the smallest feasible radius of C_R and C_B subject to their anchors is equal to the maximum of $r_{p,p'}$ among all pairs (p, p') of S.

These observations lead to a simple and efficient algorithm. First of all, H can be found in linear time, and there are $O(1)$ combinations of vertices of H to anchor C_R and C_B. Moreover, the smallest feasible radius of C_R and C_B for each anchor combination can be computed in linear time. Hence the following result is obtained.

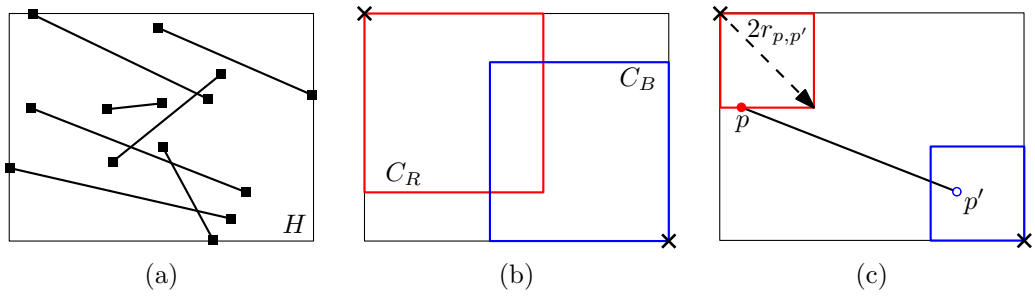
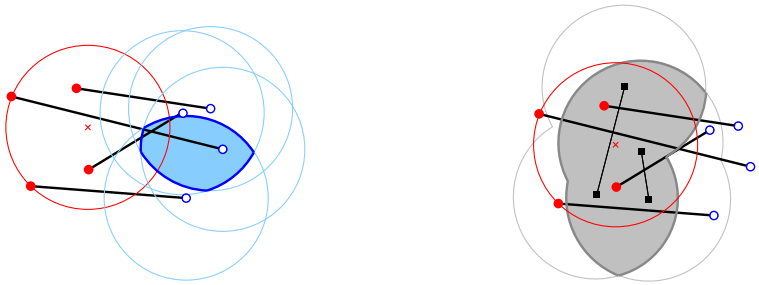


Fig. 2. (a) H is the smallest axis-aligned rectangle covering S. (b) We assume the optimal squares have vertices in corners of H. (c) For each pair (p, p') we can determine in constant time the smallest radius $r_{p,p'}$ needed to cover the pair with C_R and C_B.

Theorem 1. *The* MinMax-L_∞ *problem can be solved in optimal time* $\Theta(n)$.



(a) (b)

Fig. 3. (a) The set I_D: possible locations for centers of blue disks. (b) The set I_{DD}: intersection of all pairs of disks with both points inside C_R.

3 The MinMax-L_2 problem

3.1 An exact algorithm

We assume that optimal disks C_R and C_B have equal radius, denoted by r^*. Observe that we can further assume that one of the disks is the minimum enclosing disk of its corresponding points of S. Otherwise, it would be possible to shrink both disks by at least some small amount, contradicting the optimality of C_R and C_B.

The overall idea is to perform a binary search on all the candidate radii r for C_R and C_B, testing whether there exists a feasible solution (C'_R, C'_B) in which the radius of both disks is equal to r. Since the minimum enclosing disk of a set of n points is defined by either two or three points, there are $O(n^3)$ candidate values for r.

For each candidate radius r, we potentially test all the $\Theta(n^2)$ disks of radius r that have two points from S on its boundary. Each of those disks is a candidate for one of the disks C'_R and C'_B. Without loss of generality, we assume that it is the disk C'_R. Then we test if there exists a second disk of radius r, C'_B, that together with C'_R forms a feasible solution.

For each candidate disk C'_R, we proceed as follows. If there are pairs of S with both points outside C'_R, then C'_R is discarded as a candidate disk. Otherwise, C'_R covers at least one point of each pair. The question is then whether a feasible second disk C'_B exists. Three situations can occur.

1. Both points of each pair are inside C'_R. In this case, C'_R and $C'_B = C'_R$ form a feasible solution.
2. Each pair of S has only one point in C'_R. All these points are colored red and we can take C'_B as the minimum enclosing disk of the remaining (blue) points. There is a feasible solution for C'_R if and only if the resulting C'_B has radius at most r.
3. Otherwise, points outside C'_R are colored blue and their counterparts red. To color the remaining pairs we need a more involved procedure that is explained next.

In order to handle the third case, we start by finding the locus of the centers of the disks with radius r that cover the points outside C'_R (trivially blue). We denote this locus by I_D, which corresponds to the intersection of all disks with radius r centered at blue points (see Fig. 3(a)). The region I_D is convex, its boundary has linear complexity, and can be computed in $O(n \log n)$ time [10,26].

For a pair of points, consider the union of the two disks of radius r centered at each of the points of the pair; we call this the *(radius-r) double disk* of the pair. Let I_{DD} be the intersection of all radius-r double disks corresponding to pairs of points inside C'_R. Note that any disk with radius r centered in I_{DD} covers, at least, one point of each pair inside C'_R (see Fig. 3(b)). Using several geometric properties of both I_D and I_{DD}, we can prove the following important lemma:

Lemma 1. *A point in the intersection $I_D \cap I_{DD}$, if one exists, can be computed in $O(n \log n)$ time.*

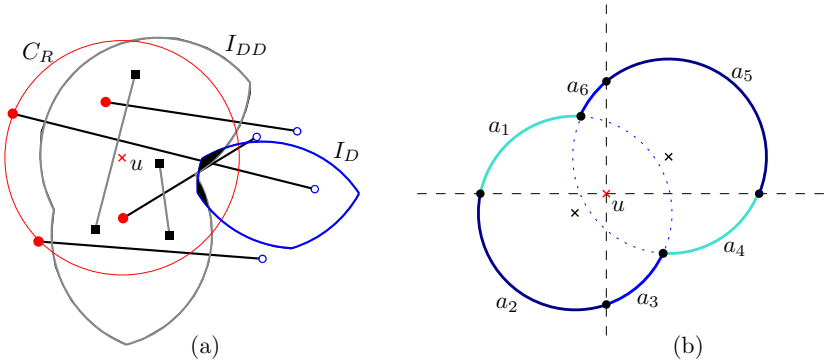


Fig. 4. (a) $I_D \cap I_{DD}$. (b) Splitting the boundary of the union of each pair of disks into six arcs $a_1, a_2, \ldots, a_6$, so that no two arcs intersect in more than one point.

Proof. We will first prove that the complexity of I_{DD} is $O(n\alpha(n))$ and can be computed in $O(n \log n)$ time, where $\alpha(n)$ is the extremely slowly-growing inverse of Ackermann's function. Second, we will show how to find a point in $I_{DD} \cap I_D$ in nearly linear time, if $I_{DD} \cap I_D \neq \emptyset$. Refer to Fig. 4(a) during the proof.

Observe that I_{DD} is a star-shaped region bounded by r-radius circular arcs, and the center point of C'_R, denoted by u, belongs to the kernel of I_{DD}. This holds because all disks that determine I_{DD} contain u. To upper bound the complexity of I_{DD} we bound the number of arcs of its boundary.

To simplify the analysis, we split the boundary of each double disk $C_1 \cup C_2$ defining I_{DD} into six arcs by using the two points in which the boundaries of C_1 and C_2 intersect, and the four intersection points of the boundary of $C_1 \cup C_2$ with the two axis-parallel lines that go through u, as shown in Fig. 4(b). Hence we obtain a set Z of at most $6n$ arcs. The two axis-parallel lines through u split the plane into four quadrants, with apex at u. The important property of this way of splitting each double disk boundary into six arcs is the following. For any two disks of equal radius C and C' whose boundaries intersect in two points a, b, if one considers any third point c in the interior of $C \cap C'$, then $\angle acb > \pi/2$. In particular, when $c = u$, this implies that the intersection points a and b belong to different (open) quadrants from u.

This property, together with the fact that u belongs to $C_1 \cap C_2$ for every double disk $C_1 \cup C_2$, implies that the two intersection points between the boundaries of every pair of different disks, each belonging to some double disk defining I_{DD}, must lie in different quadrants from u, thus the two intersection points must involve different pairs of arcs from Z. In other words, every pair of arcs taken from Z intersects at most once. Hence, computing I_{DD} is equivalent to computing the lower envelope of the set of arcs Z, seen radially around u. Since each pair of arcs intersects at most once, it follows from standard results of combinatorial geometry that I_{DD} has complexity $O(n\alpha(n))$ [32, Theorem 1.9] and can be computed in $O(n \log n)$ time [24].

It remains to compute a point in $I_D \cap I_{DD}$, where I_D has complexity $O(n)$ and I_{DD} has complexity $O(n\alpha(n))$. Since I_D is convex and I_{DD} is star-shaped with respect to u, an angular sweep around u allows to keep track of the closest arc to u for each angle, and to find a point in their intersections, if one exists, in time proportional to the complexities of I_D and I_{DD}. Altogether, the running time is $O(n \log n)$. □

If this intersection is non-empty, then there exists a point u' that is at distance at most r from all blue points (i.e. u' is in I_D), and is at distance at most r from at least one point from each pair totally inside C'_R (i.e. u' is in I_{DD}). Therefore, a disk of radius r centered at u' is a feasible candidate for C'_B. It then follows that C'_R and C'_B form a feasible solution. Note that the converse is also true: if the intersection is empty, no feasible solution for C'_R exists (however, this does not rule out that some other disk different from C'_R, with the same radius r, can be part of a feasible solution).

In summary, the algorithm has two phases. In the first phase the candidate radii are computed and sorted in $O(n^3 \log n)$ time. The second phase consists in the binary search on the radii to find the optimal value r^*. Solving an instance of the decision problem for a particular value of r takes time $O(n^2) \cdot O(n \log n) = O(n^3 \log n)$, and this is performed $O(\log n)$ times. The following result is thus obtained.

Theorem 2. *The* MinMax-L_2 *problem can be solved in* $O(n^3 \log^2 n)$ *time.*

3.2 Two approximation algorithms

In this section we present two simple algorithms that give $(1+\varepsilon)$-approximations for the MinMax-L_2 problem, for any constant $\varepsilon \in (0, 1)$. Our methods are similar to techniques used for approximating the standard k-center problem [3].

Algorithm 1: First, we apply the MinMax-L_∞ algorithm given in Section 2 and obtain squares Q_R and Q_B covering the red and blue points, respectively. Observe that the solution induced by (Q_R, Q_B) is a $\sqrt{2}$-approximation for the MinMax-L_2 problem. Assume without loss of generality that Q_R is larger than Q_B. For simplicity, scale the point set so that Q_R becomes a 1×1 square, thus it has circumradius $\sqrt{2}/2$. Let r^* denote the radius of optimal disks C_R and C_B of an optimal solution to the MinMax-L_2 problem. Then we have $1/2 \leq r^* \leq \sqrt{2}/2$. Additionally, notice that the centers of C_R and C_B lie in the bounding box of S.

Next we overlay a square grid on top of the point set (see Fig. 5). Each cell has size $\varepsilon/4 \times \varepsilon/4$. Notice that we are only interested in grid cells that potentially intersect the (unknown) optimal disks C_R and C_B. Since the centers of C_R and C_B lie in the bounding box of S, and $r^* \leq \sqrt{2}/2$, it suffices to consider the grid restricted to the bounding box of S, together with a buffer of width $\sqrt{2}/2$. In this way, the set of all cells considered, denoted by $\mathcal{C}$, has size $O(1/\varepsilon^2)$.

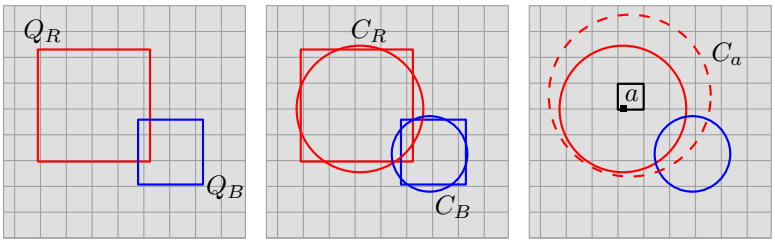


Fig. 5. Left: schematic drawing of two optimal squares and the set of cells $\mathcal{C}$ (shaded). Center: optimal disks C_R and C_B. Right: C_a (dashed) is a $(1+\varepsilon)$-approximation of C_R (solid). The center u of C_R is shown with a black square.

The algorithm performs a binary search on the radius of the optimal (largest) disk, and tests whether there exists a feasible solution (C'_R, C'_B) with the current radius. The main difference in comparison with the setting of the previous section is that, for a fixed radius, it is enough to consider only one disk for each grid cell, resulting in overall $O(1/\varepsilon^2)$ disks. Moreover, we do not try all possible radii, but only those in the set $\mathcal{R} = \{1/2, 1/2 + \varepsilon/4, 1/2 + 2\varepsilon/4, \ldots, \sqrt{2}/2\}$. Then, there are only $\Theta(1/\varepsilon)$ possible radii.

For any cell a of $\mathcal{C}$ and radius $r \in \mathcal{R}$, let $C_a(r)$ denote the disk of radius r centered at the center of the cell a. Given a fixed radius $r \in \mathcal{R}$, for each cell a of $\mathcal{C}$ we set $C'_R = C_a(r)$ and test (using the $O(n \log n)$-time algorithm of Section 3.1) if there exists a second disk C'_B of radius r such that (C'_R, C'_B) is a feasible solution. This algorithm has $O((n/\varepsilon^2) \log n \log(1/\varepsilon))$ running time. It only remains to show that it indeed computes a $(1+\varepsilon)$-approximation (see next lemma).

Lemma 2. *Let* C'_R *and* C'_B *be the disks reported by the algorithm for some* $\varepsilon \in (0, 1)$*. Then the radius of* C'_R *and* C'_B *is at most* $(1+\varepsilon)r^*$*, where* r^* *is the radius of the optimal disks.*

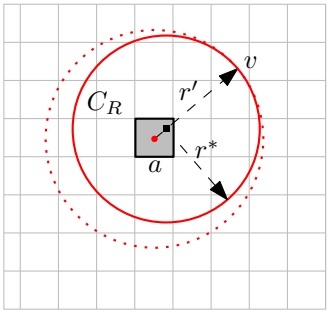


Fig. 6. The optimal disk C_R has center, shown with a black square, in cell a. The boundary dotted disk, of radius r', is the smallest disk centered at the center of a (shown with a red dot) and contains C_R.

Proof. The proof is based on the claim that the algorithm (eventually) tries a radius $\hat{r} \in \mathcal{R}$ and a cell a of $\mathcal{C}$ such that: $C_R \subseteq C_a(\hat{r})$ and $\hat{r} \leq (1+\varepsilon)r^*$.

Let u denote the center of C_R (refer to Fig. 6). By construction, the cells in $\mathcal{C}$ cover all C_R. Let a be the cell containing u, and u' denote the center of a. Let v be the intersection point between the ray emanating from u' that contains u, and the boundary of C_R. Then, the disk centered at u' and having v on its boundary contains C_R. Since u and u' lie in the same cell, the radius r' of this disk satisfies $r' \leq r^* + \varepsilon/(\sqrt{2} \cdot 4) < r^* + \varepsilon/4$. As the difference between two consecutive radii in $\mathcal{R}$ is $\varepsilon/4$, there is a radius $\hat{r} \in \mathcal{R}$ such that $r' \leq \hat{r} \leq r' + \varepsilon/4$. Therefore, $C_a(\hat{r})$ contains the optimal disk C_R. Moreover, we have that $\hat{r} \leq r^* + \varepsilon/2$. Using that $1/2 \leq r^*$, we obtain $\varepsilon/2 \leq \varepsilon r^*$, which implies $\hat{r} \leq r^* + \varepsilon r^* = (1+\varepsilon)r^*$. Since $C_R \subseteq C_a(\hat{r})$ and $r^* \leq \hat{r}$, when the algorithm tries the disk $C'_R = C_a(\hat{r})$, a second disk C'_B with radius $\hat{r}$ exists such that (C'_R, C'_B) is a feasible pair of disks for the problem, and is found by the algorithm. □

Algorithm 2 (Decoupling n and $1/\varepsilon$ in the running time): The following alternate method allows to make the dependency between n and $1/\varepsilon$ in the running time of the approximation algorithm additive, instead of multiplicative.

The square grid is the same as before, each cell has size $\varepsilon/6 \times \varepsilon/6$ and the set of $O(1/\varepsilon)$ possible radii to try in the search is $\mathcal{R}' = \{1/2, 1/2+\varepsilon/6, 1/2+2\varepsilon/6, \ldots, \sqrt{2}/2\}$. In an additional preprocessing phase, we identify all *pairs of cells* that have a pair of points using them. That is, for every pair of points of the input, we compute the corresponding grid cell for each point. For each pair of cells identified, we add the pair of their center points to the set $\mathcal{L}$. For each pair of points of S this takes $O(1)$ time (assuming the floor function is available), then $\mathcal{L}$ can be computed in $O(n)$ time using $O(1/\varepsilon^4)$ space. Note that the number N of elements of $\mathcal{L}$ satisfies $N \leq \min\{n, K/\varepsilon^4\} = O(1/\varepsilon^4)$, for some constant K.

Next we proceed (essentially) as in the exact algorithm of Section 3.1, considering only the pairs of points of $\mathcal{L}$ as input, doing the binary search on the set $\mathcal{R}'$ for the radius of C_R, and trying all the possible $\Theta(1/\varepsilon^2)$ disks C_R of that radius centered at the cell centers. For each candidate disk C_R, we identify in $O(N) = O(1/\varepsilon^4)$ time all the pairs of $\mathcal{L}$ having both points inside C_R, one inside and one outside, or both outside. If $\mathcal{L}$ contains a pair with both points outside, C_R can be immediately discarded. Otherwise, we use Lemma 1 of Section 3.1 to find C_B in $O(N \log N) = O((1/\varepsilon^4)\log(1/\varepsilon))$ time, if such a disk exists. The total running time of this approximation algorithm is then $O(n + (1/\varepsilon^2)\log(1/\varepsilon) N \log N) = O(n + (1/\varepsilon^6)\log^2(1/\varepsilon))$.

The proof that the algorithm computes a $(1+\varepsilon)$-approximation of the optimal disks goes along the same lines as the proof of Lemma 2: Let r^* be the radius of the optimal disks. Then, there exists a disk C'_R of radius $r' = r^* + \sqrt{2}(\varepsilon/6)$ that covers C_R, is centered at the center of some cell, and contains, for every point p of S in C_R, the center point of the cell that contains p. A similar disk C'_B of radius r' exists for the optimal disk C_B. The radius reported by our approximation is at least r^* and at most $r' + \varepsilon/6 = r^* + \sqrt{2}(\varepsilon/6) + \varepsilon/6 \leq (1+\varepsilon)r^*$ since it holds by construction that $1/2 \leq r^* \leq \sqrt{2}/2$. Hence, our algorithm is a $(1+\varepsilon)$-approximation.

Theorem 3. *For any $\varepsilon \in (0,1)$, a $(1+\varepsilon)$-approximation of the* MinMax-L_2 *problem can be found in $O((n/\varepsilon^2)\log n \log(1/\varepsilon))$ or $O(n + (1/\varepsilon^6)\log^2(1/\varepsilon))$ time.*

Comparing the exact algorithm, Algorithm 1, and Algorithm 2: Ignoring the log factors in the running times, we briefly make a comparison between the running times of the exact algorithm and the two approximation algorithms. Observe that the (asymptotic) running time of Algorithm 1 is (roughly) less than the $O(n^3 \log^2 n)$ running time of the exact algorithm if $\varepsilon > n^{-1}$. For Algorithm 2, the running time improves on that of the exact algorithm if $\varepsilon > n^{-1/2}$. On the other hand, Algorithm 1 has a better asymptotic running time than Algorithm 2 if $\varepsilon \leq n^{-1/4}$. Thus, we can run the exact algorithm for $\varepsilon \in (0, n^{-1}]$, run Algorithm 1 for $\varepsilon \in (n^{-1}, n^{-1/4}]$, and run Algorithm 2 for $\varepsilon \in (n^{-1/4}, 1)$.

4 The 1-center problems for pairs of points

In this section we study the 1-center problems for pairs of points, for the metrics L_∞ and L_2. Both problems are related to the Minimum Color Spanning Object problem, when the object is disk or a square, studied by Abellanas et al. [1]: Given n points in the plane, colored with $k \leq n$ colors, find a smallest axis-parallel square or disk that contains at least one point of each color. For both objects, the authors gave a solution in $O(kn \log n)$ time using the upper envelope of Voronoi surfaces [25,32]. Any instance of the 1-center problems can be reduced to an instance of the Minimum Color Spanning Object problem by coloring the $2n$ points with $k = n$ colors, so that two points receive the same color if and only if they form a pair.

In the case of the Pairs of Points L_∞ 1-Center problem, we must find a square of smallest size that contains at least one point from each pair. As the following theorem shows, an optimal square can be found in nearly linear time.

Theorem 4. *The* Pairs of Points L_∞ 1-Center *problem can be solved in $O(n \log^2 n)$ time in the worst case.*

Proof. We first consider the *decision version* of the problem: Given a size $d > 0$, does there exist a square of size d (i.e. radius $d/2$) covering at least one point of each pair? This question can be answered in $O(n \log n)$ time as follows.

For each point p of S, let H_p be the axis-aligned square of size d centered at p. Given paired points p and p' of S, represent the set $H_p \cup H_{p'}$ by the union of at most three rectangles with pairwise disjoint interiors. Let $Q_{p,p'}$ denote the set of those rectangles. Then the problem reduces to asking if the depth of the arrangement induced by the union of the sets $Q_{p,p'}$, over all paired points p and p' of S, is equal to n. This can be solved in $O(n \log n)$ time (see for example [6])[9].

Next we can integrate the solution of the decision problem into an optimization algorithm, based on matrix searching.

First notice that there always exists an optimal solution Q having points of S in two opposite sides, and the L_∞ distance between those points is the size of Q. Then, every two points p and q of S determine at most two values for the parameter d, $|x(p) - x(q)|$ and $|y(p) - y(q)|$. In order to compute the optimal value for d among all candidates of the form $|x(p) - x(q)|$ (for two points p, q of S), we do the following: Let $p_1, p_2, \ldots, p_{2n}$ be the points of S sorted by x-coordinate in ascending order, and consider the $2n \times 2n$ matrix M such that:

$$M_{i,j} = \begin{cases} x(p_i) - x(p_{2n-j+1}) & \text{if } i > 2n - j + 1 \\ (i+j) - 2n - 2 & \text{if } i \leq 2n - j + 1 \end{cases}$$

[9] Notice that, if a point q in the plane belongs to the boundary of two rectangles in $Q_{p,p'}$ and these two rectangles come from the subdivision of the same set $H_p \cup H_{p'}$, then we need that only one of two rectangles contributes to the depth at q, in order for our characterization to be correct. The algorithm in [6] can be adapted to measure the depth in the desired way for this particular case.

Note that M is a sorted matrix (i.e. every row and every column is sorted) containing all the possible values of d of the form $|x(p) - x(q)|$. Hence we can apply matrix searching [4] in order to find the optimal value of d, solving $O(\log n)$ instances of the decision problem. Finally, we obtain an $O(n \log^2 n)$-time algorithm since the value of every entry of M can be computed in constant time, once we know the order of S by x-coordinate.

We use an analogous procedure to compute the optimal value for d among all candidates of the form $|y(p) - y(q)|$. □

In the following we show that the Pairs of Points L_∞ 1-Center problem can be solved in $O(n \log n)$ expected time by using the Chan's randomized technique for optimization problems [13]. After that, we complement this result by proving that this problem has an $\Omega(n \log n)$ lower bound in the algebraic decision tree model.

Lemma 3 (Lemma 2.1 in [13]). *Let $\alpha < 1$, $\varepsilon > 0$, and r be constants, and let $D(\cdot)$ be a function such that $D(n)/n^\varepsilon$ is monotone increasing in n. Given any problem P of size n and solution $w(P)$, suppose that within $D(n)$ time,*

(a) we can decide whether $w(P) < t$ for any given $t \in \mathbb{R}$, and
(b) we can construct r subproblems $P_1, \ldots, P_r$, each of size at most $\lceil \alpha n \rceil$, so that

$$w(P) = \min\{w(P_1), \ldots, w(P_r)\}.$$

Then for any problem P, we can compute the solution $w(P)$ in $O(D(n))$ expected time.

Theorem 5. *The* Pairs of Points L_∞ 1-Center *problem can be solved in $O(n \log n)$ expected time.*

Proof. The idea is to use Lemma 3, specifically an adaptation of Chan's arguments to solve in $O(n \log n)$ expected time the problem of finding a smallest axis-parallel square that contains at least k of n given points [13].

We extend the problem slightly: given a set S of n pairs of points in the plane and a rectangle T, we will compute $w(S, T)$, the radius of the smallest square that contains at least one element from each pair of S and, in addition, contains T (initially, T is empty). The $O(n \log n)$-time decision algorithm in the proof of Theorem 4 can be modified for this extended problem. Further, in linear time we can divide the problem into at most five subproblems by using the next 4 lines: Let ℓ_1 be the leftmost line between the next two lines: (1) the vertical line at the $\lceil n/5 \rceil$th smallest x-coordinate of the points in S, and (2) the leftmost vertical line at the right point of a pair of S. We say that ℓ_1 is an *extreme* line if and only if it falls in the case (2). Symmetrically, we can define the vertical rightmost line ℓ_2, the horizontal bottommost line ℓ_3, and the horizontal topmost line ℓ_4. Let T_0 denote the rectangle bounded by ℓ_1, ℓ_2, ℓ_3, and ℓ_4, and let H_i $(i = 1, 2, 3, 4)$ denote the open halfplane bounded by ℓ_i that contains the interior of T_0. The optimal square Q must belong to one of the next two cases:

Case 1: Q contains T_0. Then $w(S, T) = w(S', T + T_0)$, where S' is the set of the pairs of S that have both points not in T_0, and $T + T_0$ denotes the minimum enclosing rectangle of $T \cup T_0$.

Case 2: $Q \subset H_i$ for some $i \in \{1, \ldots, 4\}$ where ℓ_i is not an extreme line. Then $w(S, T) = w(S'', T + T_i)$, where S'' is the set of the pairs of S that have both points in H_i, and T_i is the minimum enclosing rectangle of the partners of the points not in H_i.

The first case generates one subproblem of size at most $2\lceil n/5 \rceil$ and the second one generates at most four subproblems each of size at most $4\lceil n/5 \rceil$. Lemma 3 can thus be applied with $\alpha = 4/5$, $\varepsilon = 1$, and $r = 5$. □

Theorem 6. *The* Pairs of Points L_∞ 1-Center *problem has an $\Omega(n \log n)$ lower bound in the algebraic decision tree model, even in one dimension.*

Proof. Given a set $X = \{x_1, x_2, \ldots, x_n\}$ of n real numbers, $\mathsf{MAXGAP}(X)$ (i.e. the maximum gap of X) is defined as the maximum difference between consecutive elements of X. Computing $\mathsf{MAXGAP}(X)$ is known to have an $\Omega(n \log n)$ lower bound in the algebraic decision tree model [29].

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of n real numbers, and assume w.l.o.g. that $0 < x_i < 1$ for every i. Let $x'_1 < x'_2 < \cdots < x'_n$ be the sorting of $x_1, x_2, \ldots, x_n$. We build the following instance of the Pairs of Points L_∞ 1-Center problem on the real line: for $i = 1, \ldots, n$ we add the pair of points $-1 + x_i, x_i$. Finally, we add the pairs $-1 + x'_n, -1 + x'_n$ and x'_1, x'_1. Let $[\alpha, \beta]$ be a solution of this instance. Observe that $\alpha = -1 + x'_i$ and $\beta = x'_{i-1}$ for some $i \in [2 \ldots n]$. Furthermore, we have that $\beta - \alpha = 1 - (x'_i - x'_{i-1})$, which implies $\mathsf{MAXGAP}(X) = 1 - (\beta - \alpha)$. Since this construction can be done in $O(n)$ time, the theorem thus follows via problem reduction. □

In one dimension, the Pairs of Points L_∞ 1-Center problem can be solved in $O(n)$ time if the points are already sorted. Namely, the problem can be reduced in linear time to an instance of the following modified problem: Given an interval s of the real line and a set S of n pairs of points such that each pair of S has one element to the left of s and the other element to the right of s, compute the solution of the Pairs of Points L_∞ 1-Center problem for S subject to contain s. We leave the details as an exercise.

The above results for the Pairs of Points L_∞ 1-Center problem can be straightforward modified to solve the Minimum Color Spanning Object problem of Abellanas et al. [1] when the object is a square. Comparing above results with their $O(kn \log n)$-time algorithm, our worst-case $O(n \log^2 n)$-time algorithm represents an improvement for $k = \omega(\log n)$, and the $O(n \log n)$-time expected algorithm reduces the complexity in a factor of k.

For the Pairs of Points L_2 1-Center problem the goal is to find a disk of smallest radius containing at least one point from each pair. As expected, this problem seems to be harder than its square counterpart. Note that Theorem 6 implies an $\Omega(n \log n)$ lower bound for the Pairs of Points L_2 1-Center problem.

Theorem 7. *The* Pairs of Points L_2 1-Center *problem can be solved in* $O(n^2 \log n)$ *time.*

Proof. Color the $2n$ points with $k = n$ colors, such that two points receive the same color if and only if they form a pair. Then apply the algorithm of Abellanas et al. [1] running in $O(kn \log n) = O(n^2 \log n)$ time. □

5 The MinSum-L_∞ problem

In this section we study the problem of finding two squares under the MinSum criterion, thus we want to minimize the sum of their radii. A major difference with the previous MinMax version is that now we need to take both radii into account. Let C_R and C_B denote, respectively, the two squares of an optimal solution, where C_R encloses all red points and C_B encloses all blue points. Up to symmetry, there are four relative positions of C_R and C_B, as depicted in Fig. 7. In the following, we will show how to find optimal solutions of type (a), (b), or (c). In the case in which the solution is of type (d), C_R is a minimum enclosing square of all points of S, and C_B is a solution to the Pairs of Points L_∞ 1-Center problem. Hence, this case can be solved in $O(n \log^2 n)$ time in the worst case (see Theorem 4) or in $O(n \log n)$ expected time (see Theorem 5).

In what follows we focus on cases (a)–(c). Let H be the smallest axis-aligned rectangle covering S, and p_1, p_2, p_3 be the points of S contained on the left, top, and right boundaries of H, respectively (possibly $p_1 = p_2$ or $p_2 = p_3$). We assume that p_1 and p_2 belong to the left and top boundaries of C_R, respectively. Hence, the top-left corner of H is also the top-left corner of C_R, denoted u. Further, we assume that p_3 and the bottommost point colored blue are on the right and bottom boundaries of C_B, respectively. Let v be the bottom-right vertex of C_B. By the previous assumption, $x(v) = x(p_3)$, but $y(v)$ is unknown. Therefore, vertex u is fixed (given by p_1 and p_2), but from the coordinates of the vertex v only $x(v)$ is known.

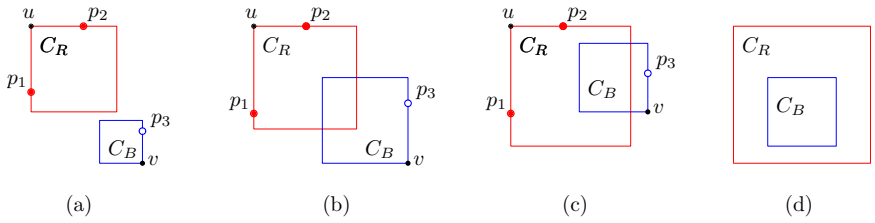


Fig. 7. Relative positions of C_R and C_B.

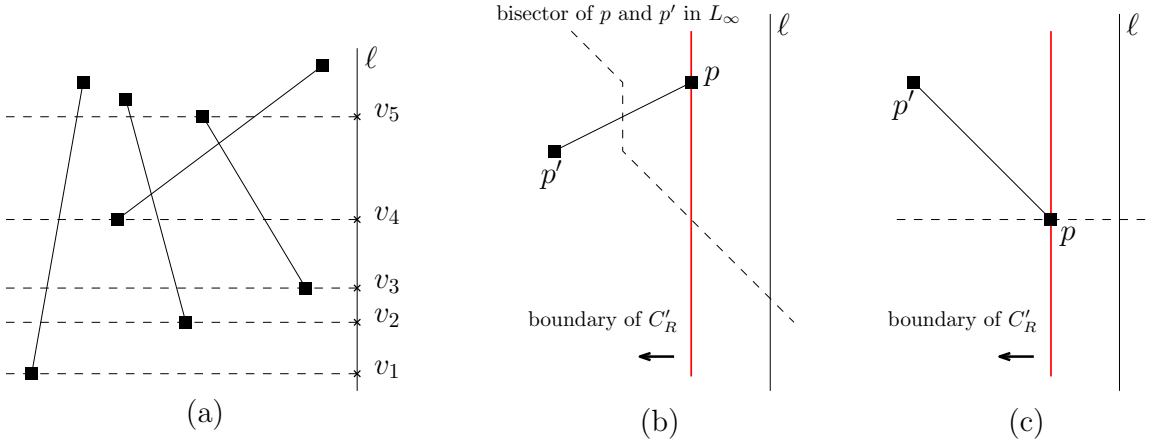


Fig. 8. (a) The possible positions $v_1, \ldots, v_5$ for vertex v. (b) One sweep event case. (c) The other sweep event case.

Algorithm overview. Initially, all points of S are considered *black*, meaning that their (red/blue) colors are undefined. We say that a pair of points of S is *black* if its two points are black.

We start with a square C'_R covering S and with its top-left vertex anchored at u. We color both p_1 and p_2 red and their partners (in their pairs) blue, and also color p_3 blue and its partner red. Next we sweep S with the boundary of C'_R by moving its bottom-right vertex (diagonally) towards u. The sweep events occur when the boundary of C'_R crosses a point p. If p is black, then we color point p blue and its partner red, and, considering C'_R fixed, compute the smallest feasible blue square C'_B. Notice that C'_B, having bottom-right vertex v, covers all points colored blue and for each black pair the point closer to v that is not lying below v. At this point the pair (C'_R, C'_B) is considered a candidate solution to our problem. If p is red, this means that C'_R cannot be made smaller, and the sweep ends. During the sweep, we keep track of the candidate solution (C'_R, C'_B) minimizing the sum of the radii.

The challenging part of the algorithm is to recompute the optimal C'_B each time the coloring changes, more precisely, to determine $y(v)$ after each event. Observe that if at any time the bottom boundary of C'_R crosses a point p of S, which must be the lowest point of S, then p will be colored blue and from that point on the vertex v will become fixed at the bottom-right corner of H. Once v is fixed, it is easy to implement the sweep in $O(n \log n)$ time. Hence, the challenging part of the algorithm is in the first part of the sweep, until the bottom boundary of C'_R finds the first point (note that, depending on the points of S, this may never happen). This implies that we can focus only on the events corresponding to points of S being crossed by the right boundary of C'_R.

Details of the algorithm. We now proceed to explain how the above sweep (only for the right boundary of C'_R) can be done in $O(n \log n)$ time. First note that there exist $O(n)$ possible locations for vertex v because the bottom boundary of optimal C_B contains the lowest point colored blue. For each candidate v there is no pair of points of S whose two elements are below v (see Fig. 8(a)).

Let $v_1, v_2, \ldots, v_m$ denote from bottom to top the possible positions for vertex v. At the kth event of the sweep ($k \geq 0$, where $k = 0$ means that no event has occurred yet), or just at event k, let $r_k(v_i)$, $i \in [1 \ldots m]$, denote the size of the smallest feasible square C'_B having v_i as bottom-right vertex. Let $C_k(v_i)$ denote such a square.

After each event $k \geq 1$, when the right boundary of C'_R crosses a black point p of S, the black pair consisting of p and its partner p' (which is to the left of p) is colored. Then the smallest square among $C_k(v_1), C_k(v_2), \ldots, C_k(v_m)$ might change and we must (implicitly) produce the values $r_k(v_1), \ldots, r_k(v_m)$ from $r_{k-1}(v_1), \ldots, r_{k-1}(v_m)$, and compute $\min\{r_k(v_1), \ldots, r_k(v_m)\}$. There are two event cases to follow according to the relative position of p and its partner p':

Case 1: p' is below p (Fig. 8(b)). We must update the $r_{k-1}(\cdot)$ value of all points v_i among $v_1, v_2, \ldots, v_m$ that are below the bisector of p and p' (in the L_∞ metric) and *do not contain* the point p. This must be done since, after this event, the square $C_k(v_i)$ must cover the point p that is colored blue and further from v_i than is p' (Fig. 9).

Case 2: p' is above or horizontally aligned with p (Fig. 8(c)). We must discard the points $v_t, v_{t+1}, \ldots, v_m$ strictly above the horizontal line through p because now p is blue and p' red. Any v_i can be discarded by considering $r_k(v_i) = +\infty$.

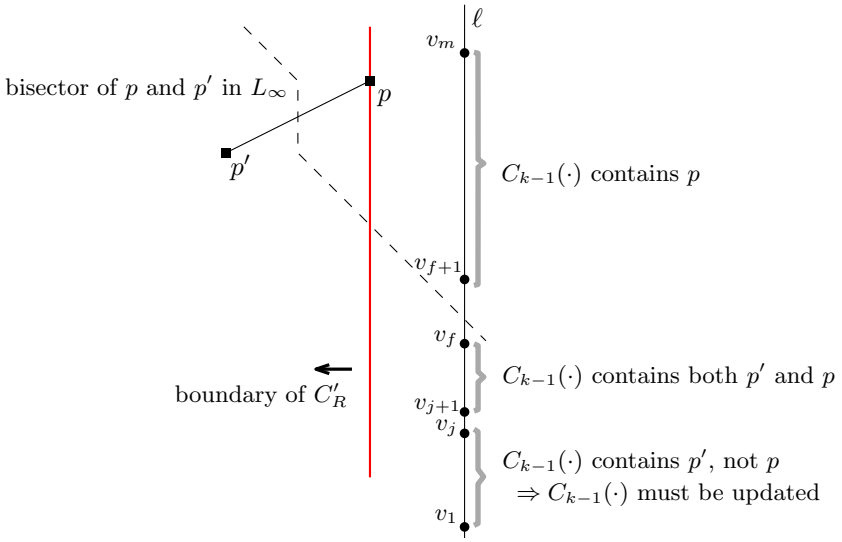


Fig. 9. The values of $r_{k-1}(\cdot)$ that need to be updated at event k, for the Case 1 in Fig. 8(b).

Lemma 4. *For every sweep event $k \geq 0$, the top boundaries of squares $C_k(v_1), \ldots, C_k(v_m)$ are sorted, that is, $y(v_1) + r_k(v_1) \leq y(v_2) + r_k(v_2) \leq \cdots \leq y(v_m) + r_k(v_m)$.*

Proof. Suppose, for the sake of contradiction, that there exists a pair $i < j$ such that $y(v_i) + r_k(v_i) > y(v_j) + r_k(v_j)$. Let p be the point in $C_k(v_i)$ at maximum distance from v_i; note that p is on the top or left side of $C_k(v_i)$. Since $y(v_i) < y(v_j)$ and $y(v_i) + r_k(v_i) > y(v_j) + r_k(v_j)$, p is not contained in $C_k(v_j)$. Hence, the partner p' of p belongs to $C_k(v_j)$. Since $C_k(v_j)$ is contained in $C_k(v_i)$, then $C_k(v_i)$ could be shrunk by covering p' instead of p, which is a contradiction. □

Lemma 5. *Let $k \geq 1$ be an event of* Case 1*, generated by the point p, and $j = j(k)$ be the largest index such that $C_{k-1}(v_j)$ does not contain p. Then the next statements are satisfied:*

(a) $C_{k-1}(v_i)$ does not contain p for $i = 1, \ldots, j$.
(b) j is the greatest index such that $y(v_j) + r_{k-1}(v_j) < y(p)$.

(c) $r_k(v_i) = y(p) - y(v_i)$ *for* $i = 1, \ldots, j$ *and* $r_k(v_i) = r_{k-1}(v_i)$ *for* $i = j+1, \ldots, m$.
(d) *For every event* $k' > k$ *of* CASE 1 *it holds* $j(k') \geq j$.

Proof. Since $C_{k-1}(v_j)$ does not contain p but does contain the partner of p, v_j is below the line with slope -1 through p, which implies $y(v_i) + r_k(v_i) < y(p)$ for $i = 1, \ldots, j$ by Lemma 4. Hence, statements (a) and (b) follow. For $i = 1, \ldots, j$, $C_{k-1}(v_i)$ must be enlarged to cover the point p, then $r_k(v_i) = y(p) - y(v_i)$ and statement (c) holds. Statement (d) can be observed from Lemma 4 and the fact that $y(v_i) + r_k(v_i) = y(p)$ for $i = 1, \ldots, j$. □

Lemma 6. *Let* $k_1 \geq 1$ *be the last event of* CASE 1*, generated by the point* p*, and let* $k_2 \geq 1$ *be the last event of* CASE 2*. Further, let* $k = \max\{k_1, k_2\}$ *be the last event among those ones. Suppose that* $v_1, v_2, \ldots, v_j$ *were updated at* k_1 *and* $v_t, v_{t+1}, \ldots, v_m$ *are all the vertices that have been discarded during the sweep up to event* k*. Then we have:*

$$\min\Big\{r_k(v_1), \ldots, r_k(v_m)\Big\} = \min\Big\{y(p) - y(v_j), r_0(v_{j+1}), r_0(v_{j+2}), \ldots, r_0(v_{t-1})\Big\}. \tag{1}$$

Proof. By Lemma 5(c), we have $r_k(v_i) = y(p) - y(v_i)$ for $i = 1, \ldots, j$, then $\min\{r_k(v_1), \ldots, r_k(v_j)\} = y(p) - y(v_j)$. By Lemma 5(d), it holds $r_k(v_i) = r_0(v_i)$ for $i = j+1, \ldots, t-1$. Since $v_t, v_{t+1}, \ldots, v_m$ have been discarded, then $r_k(v_i) = +\infty$ for $i = t, \ldots, m$. The result follows. □

Lemma 7. *Computing* $r_0(v_i)$ *for all* $i \in [1 \ldots m]$ *can be done in* $O(n \log n)$ *time.*

Proof. We start by computing $r_0(v_m)$, which can be done in linear time by finding, for each black pair, the point not below v_m that is closer to v_m. Then, we compute $r_0(v_{m-1}), r_0(v_{m-2}), \ldots, r_0(v_1)$ as follows:

Let ℓ be the vertical line containing $v_1, v_2, \ldots, v_m$; that is, ℓ is the vertical line through the rightmost point of S (assumed to be p_3). We sweep along ℓ, with a point w, from v_m to v_1 in the following way. For each position of w, we consider the n critical points of S which are those that must be covered by the smallest feasible blue square C'_B with bottom-right vertex w denoted by $C'_0(w)$. Each pair of S determines one critical point: if both points are colored, then the blue one is the critical point. Otherwise, if the pair is black, then the critical point is the element of the pair with y-coordinate greater than or equal to w's that minimizes the (L_∞) distance to w.

During the sweep we maintain a priority queue Π over the critical points, where the priority of a point is its distance to w. We denote by P_x (respectively, P_y) the set of critical points whose distance to w is realized by a difference in x-coordinates (respectively, y-coordinates). At any time, $C'_0(w)$ is determined by the critical point with maximum priority.

Priorities in Π are not explicitly maintained. Instead, we use two independent priority queues Π_x and Π_y for the sets P_x and P_y, respectively. The priority of a point $p \in P_x$ in Π_x is its distance to w, while the priority of a point $p \in P_y$ in Π_y is $y(p) - y(v_m)$. Then, if r_x is the maximum priority of Π_x and r_y is the maximum priority of Π_y, then the maximum priority of Π (i.e., the size of $C'_0(w)$) is equal to $\max\{r_x, r_y + y(v_m) - y(w)\}$. Queues Π_x and Π_y change whenever point w crosses an intersection point of ℓ with: (*i*) a line with slope -1 through a point p of S, (*ii*) a horizontal line through a point p of S, or (*iii*) the bisector in the L_∞ metric of a pair (p, p') of S.

In case (*i*) we proceed as follows. If p is not stored in Π then we continue with the sweep. Otherwise, we have that p is stored in Π_x and then we remove p from Π_x and insert p in Π_y with priority $y(p) - y(v_m)$.

In case (*ii*), if p' (which satisfies $y(p') > y(w)$ and is stored in Π) is further from w than p then we remove p' from the queue and insert p in Π_x with priority $x(p) - x(w)$.

In case (*iii*), assuming, without loss of generality, that p is the critical point before the intersection with the bisector (notice that p must be stored in Π_y), we remove p from Π_y and insert p' in Π_x with priority $x(w) - x(p')$.

It is thus easy to see now that this sweep can be done in $O(n \log n)$ time, allowing computation of $r_0(v_{m-1}), r_0(v_{m-2}), \ldots, r_0(v_1)$, in overall $O(n \log n)$ time. □

Lemma 8. *At every event* $k \geq 1$*, the square of minimum size among* $C_k(v_1), C_k(v_2), \ldots, C_k(v_m)$ *can be found in* $O(\log n)$ *time.*

Proof. Suppose that k falls in Case 1 and was generated by the point p. In the more general case, suppose that $k_1 < k$ was the previous event of Case 1, generated by the point p_1, in which the interval $v_1, v_2, \ldots, v_{j'}$ was updated. Further, suppose that $v_t, v_{t+1}, \ldots, v_m$ were discarded at events of Case 2. By Lemma 5, the value of $j \geq j'$ such that the interval $v_1, v_2, \ldots, v_j$ must be updated at event k can be found with a binary search over the values $y(p_1) < y(v_{j'+1}) + r_0(v_{j'+1}) \leq y(v_{j'+2}) + r_0(v_{j'+2}) \leq \cdots \leq y(v_{t-1}) + r_0(v_{t-1})$. By Lemma 6, we compute $\min\{r_k(v_1), \ldots, r_k(v_m)\}$ by finding the minimum value among $r_0(v_{j+1}), r_0(v_{j+2}), \ldots, r_0(v_{t-1})$, which can be done in $O(1)$ time by preprocessing $r_0(v_1), r_0(v_2), \ldots, r_0(v_m)$ for range minimum queries [8,23] or in $O(\log n)$ time using a balanced binary tree. If k falls in Case 2 the arguments are similar. By keeping track of both the point p and index j of the last event of Case 1, and the index t, each sweep event can be processed in $O(\log n)$ time. □

Putting all the above arguments together, we have that an optimal solution to the MinSum-L_∞ problem satisfying case (a), (b), or (c) can be found in $O(n \log n)$ time. The bottleneck of the running time of the algorithm comes from case (d), which requires $O(n \log^2 n)$ time in the worst case (Theorem 4) or $O(n \log n)$ expected time (Theorem 5).

Theorem 8. *The* MinSum-L_∞ *problem can be solved in* $O(n \log^2 n)$ *time in the worst case and in* $O(n \log n)$ *expect time.*

Theorem 9. *The* MinSum-L_∞ *problem has an* $\Omega(n \log n)$ *lower bound in the algebraic decision tree model.*

Proof. We use an extension of the proof of Theorem 6. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of n real numbers, and assume w.l.o.g. that $0 < x_i < 1$ for every i. Let $x'_1 < x'_2 < \cdots < x'_n$ be the sorting of $x_1, x_2, \ldots, x_n$. We build the following instance of the MinSum-L_∞ problem. On the real line we add the same points as in the proof of Theorem 6: for $i = 1, \ldots, n$ we add the pair of points $-1 + x_i, x_i$, and the pairs $-1 + x'_n, -1 + x'_n$ and x'_1, x'_1. Let S_1 denote these added points. Finally, we add four extra pairs of points in $\mathbb{R}^2$: $(2,2),(0,0)$; $(-2,2),(0,0)$; $(-2,-2),(0,0)$; and $(2,-2),(0,0)$. The repetitions of the point $(0,0)$ can be avoided with a suitable perturbation. Let $S_4 = \{(2,2),(-2,2),(-2,-2),(2,-2)\}$. If each square of the solution covers at least two elements of S_4, then the solution has value 4. Otherwise, if one square contains three elements of S_4 and the other one contains exactly one, then the solution has value at least 3, since the second square must contain the origin $(0,0)$. Therefore, in the optimal solution, one square contains all S_4 (and thus all points) and the other square (which must contain the origin $(0,0)$) contains from each of the pairs in S_1 at least one point, case in which the solution has value less than $5/2$. Then, the second square is the solution of the Pairs of Points L_∞ 1-Center problem for S_1. Therefore, using the same arguments as in the proof of Theorem 6, it follows that by solving this instance of the MinSum-L_∞ problem we can compute $\mathsf{MAXGAP}(X)$. □

6 The MinSum-L_2 problem

6.1 Exact algorithms

This variant of the problem seems to be considerably harder than the previous ones. A brute force approach tries each possible pair of disks defined by points in S, and checks each pair for feasibility, leading to $O(n^7)$ running time. In this section we present a faster algorithm, which unfortunately still has a rather high running time.

The MinSum-L_2 problem can be solved by considering each possible disk C_R that contains at least one point from each pair. For each selection of C_R, the pairs with one point in C_R and one outside become colored accordingly, while for the others the color assignment is still unclear. Then we must compute the minimum enclosing disk C_B of all blue points and at least one point of each uncolored pair. It is easy to see that the computation of C_B corresponds to an instance of the Pairs of Points L_2 1-Center problem, and can thus be solved in $O(n^2 \log n)$ time (by Theorem 7). This implies an overall $O(n^5 \log n)$-time algorithm. Notice that any improvement in

the time needed to solve the PAIRS OF POINTS L_2 1-CENTER problem would result in a faster algorithm for the MINSUM-L_2 problem.

A faster alternative is to follow the lines of the algorithm for the MINMAX-L_2 problem in Section 3.1. We try all possible disks for C_R, that is, $\Theta(n^3)$ of them. For each fixed C_R, we find the smallest possible C_B for the chosen C_R, as follows.

Assume without loss of generality that C_B is larger (or equal) than C_R. We use the same approach as in Section 3.1: we compute I_D and I_{DD}. The only difference is that, since we do not know the radius of C_B, we need an extra binary search to find it. Because we assumed that C_B is larger than C_R, I_{DD} is still star-shaped from the center of C_R. Then we can still compute a point in the intersection of I_D and I_{DD} in $O(n \log n)$ time, in exactly the same way. This results in an $O(n^4 \log^2 n)$-time algorithm.

6.2 Approximation schemes

Given the high running time of our exact algorithms, it is worth noting that $O(n)$-time approximations very similar to the ones of Section 3.2 can be applied to this problem as well. There are two main differences.

On the one hand, we need a different initial constant-factor approximation. We can, again, use the algorithm for the MINMAX-L_∞ problem of Section 2 to compute the initial approximation. It is not hard to verify that the solution to the MINMAX-L_2 problem is a 2-approximation for the MINSUM-L_2 problem. Therefore, the solution obtained with the algorithm of Section 2 gives an initial $2\sqrt{2}$-approximation for the MINSUM-L_2 problem. Adjusting the size of the grid cells accordingly, a very similar approach leads to a $(1+\varepsilon)$-approximation algorithm.

On the other hand, we cannot use a binary search to guess the radius of the first disk, as we do in the first approximation algorithm of Section 3.2. Instead, we have to try all possible disks for C_R, of which there are $O(1/\varepsilon^3)$ ones. Once C_R is fixed, we can do a binary search on the radius of C_B (there are $O(1/\varepsilon)$ radii and $O(1/\varepsilon^2)$ disks for each radius), to find the smallest possible C_B for the choice of C_R, in $O((n/\varepsilon^2)\log(1/\varepsilon))$ time. Therefore, this leads to a first $(1+\varepsilon)$-approximation running in $O((n/\varepsilon^5)\log(1/\varepsilon))$ time.

The second approximation in Section 3.2 that has a running time of the form $O(n+\text{polylog}(1/\varepsilon))$ can be adapted for the MINSUM-L_2 problem. Again, the initial binary search cannot be used, and finding C_R takes $O(1/\varepsilon^3)$ time instead of $O(1/\varepsilon^2)$. This leads to a second $(1+\varepsilon)$-approximation running in $O(n + (1/\varepsilon^7)\log^2(1/\varepsilon))$ time.

We summarize the results in this section with the following theorem.

Theorem 10. *The* MINSUM-L_2 *problem can be solved in* $O(n^4 \log^2 n)$ *time. A* $(1+\varepsilon)$*-approximation can be computed in* $O((n/\varepsilon^5)\log(1/\varepsilon))$ *or* $O(n + (1/\varepsilon^7)\log^2(1/\varepsilon))$ *time for any* $\varepsilon \in (0,1)$.

Comparing the asymptotic running times of the exact algorithm and the two approximation algorithms (ignoring the log factors), we can run the exact algorithm for $\varepsilon \in (0, n^{-3/5}]$, run the first approximation algorithm in $O((n/\varepsilon^5)\log(1/\varepsilon))$ time for $\varepsilon \in (n^{-3/5}, n^{-1/2}]$, and run the second approximation algorithm in $O(n + (1/\varepsilon^7)\log^2(1/\varepsilon))$ time for $\varepsilon \in (n^{-1/2}, 1)$.

7 Further research

The main open problems derived from this work are related to improving several of our algorithms, in particular the solutions given for MINMAX-L_2 problem, MINSUM-L_2 problem, and PAIRS OF POINTS L_2 1-CENTER problem. In particular, it is an intriguing open question whether there exists a subquadratic time algorithm for the later problem. On the other hand, it would also be interesting to obtain lower bounds for these three problems, and to study all of our problems in higher dimensions.

Acknowledgments. We thank Mercè Claverol for participating in early versions of this work, and all other participants of the 7th Iberian Workshop on Computational Geometry, partially funded by HP2008-0060, for helpful discussions. E. A. and J. M. were partially supported by the National Science Foundation (CCF-1018388), the US-Israel Binational Science Foundation (BSF 2010074), Metron Aviation (subcontract from NASA Ames), and Sandia National Labs. J.M. D.-B. was partially supported by project FEDER MEC MTM2009-08652. J.M. D.-B., F. H., and R.I. S. were partially supported by ESF EUROCORES programme EuroGIGA, CRP ComPoSe: grant EUI-EURC-2011-4306. F. H., B. P., and R.I. S. were partially supported by project MINECO MTM2012-30951. F. H. and R.I. S. were partially supported by project Gen. Cat. DGR 2014SGR46. F. H. and M. S. were partially supported by projects MTM2009-07242 and Gen. Cat. DGR 2009SGR1040. P. K. was partially supported by National Science Foundation through CAREER Grant CCF-0643593, and the Air Force Young Investigator Program. B. P. was partially supported by project VA172A12-2. P. P.-L. was partially supported by projects FEDER MEC MTM2009-08652, CONICYT FONDECYT/Iniciación 11110069 (Chile), and Millennium Nucleus Information and Coordination in Networks ICM/FIC P10-024F (Chile). M. S. was funded by projects NEXLIZ - CZ.1.07/2.3.00/30.0038, which is co-financed by the European Social Fund and the state budget of the Czech Republic, ESF EuroGIGA project ComPoSe as F.R.S.-FNRS - EUROGIGA NR 13604, and ESF EuroGIGA project GraDR as GAČR GIG/11/E023. R.I. S. was funded by Portuguese funds through CIDMA (Center for Research and Development in Mathematics and Applications) and FCT (Fundação para a Ciência e a Tecnologia), within project PEst-OE/MAT/UI4106/2014, and by FCT grant SFRH/BPD/88455/2012.

References

1. M. Abellanas, F. Hurtado, C. Icking, R. Klein, E. Langetepe, L. Ma, B. Palop, and V. Sacristán. Smallest color-spanning objects. In *ESA 2001: Algorithms*, volume 2161 of *LNCS*, pages 278–289. 2001.
2. M. Abellanas, F. Hurtado, C. Icking, R. Klein, E. Langetepe, L. Ma, B. Palop, and V. Sacristán. Voronoi diagram for services neighboring a highway. *Information Processing Letters*, 86:283–288, 2003.
3. P. K. Agarwal and C. M. Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002.
4. P. K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. *ACM Computing Surveys*, 30(4):412–458, 1998.
5. H.-K. Ahn, H. Alt, T. Asano, S. W. Bae, P. Brass, O. Cheong, C. Knauer, H.-S. Na, C.-S. Shin, and A. Wolff. Constructing optimal highways. *International Journal of Foundations of Computer Science*, 20(1):3–23, 2009.
6. H. Alt and L. Scharf. Computing the depth of an arrangement of axis-aligned rectangles in parallel. In *Abstracts of the 26th European Workshop on Computational Geometry (EuroCG)*, pages 33–36, 2010.
7. R. Batta and U. S. Palekar. Mixed planar/network facility location problems. *Computers & Operations Research*, 15:61–67, 1988.
8. M. A. Bender and M. Farach-Colton. The LCA problem revisited. In *LATIN 2000: Theoretical Informatics*, volume 1776 of *LNCS*, pages 88–94. 2000.
9. S. Bespamyatnikh and M. Segal. Rectilinear static and dynamic discrete 2-center problems. In *WADS 1999: Algorithms and Data Structures*, volume 1663 of *LNCS*, pages 276–287. 1999.
10. K. Q. Brown. *Geometric Transforms for Fast Geometric Algorithms*. PhD thesis, Department of Computer Science, Carnegie-Mellon University, 1979.
11. J. Cardinal, S. Collette, F. Hurtado, S. Langerman, and B. Palop. Optimal location of transportation devices. *Computational Geometry: Theory and Applications*, 41:219–229, 2008.
12. J. Cardinal and S. Langerman. Min-max-min geometric facility location problems. In *Abstracts of the 22nd European Workshop on Computational Geometry (EuroCG)*, pages 149–152, 2006.
13. T. M. Chan. Geometric applications of a randomized optimization technique. *Discrete & Computational Geometry*, 22:547–567, 1999.
14. T. M. Chan. More planar two-center algorithms. *Computational Geometry: Theory and Applications*, 13(3):189–198, 1999.
15. B. Chazelle and J. Matousek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *Journal of Algorithms*, 21(3):579–597, 1996.
16. M. Consuegra, G. Narasimhan, D. Rodriguez, and S. Tanigawa. Avatar problems. Technical Report TR-2012-10-06, Florida International University, 2012.

17. J. M. Díaz-Báñez, M. Korman, P. Pérez-Lantero, A. Pilz, C. Seara, and R. I. Silveira. New results on stabbing segments with a polygon. In *CIAC 2013: Algorithms and Complexity*, volume 7878 of *LNCS*, pages 146–157. 2013.
18. J. M. Díaz-Báñez, M. Korman, P. Pérez-Lantero, and I. Ventura. The 1-median and 1-highway problem. *European Journal of Operational Research*, 225(3):552–557, 2013.
19. H. Ding and J. Xu. Solving the chromatic cone clustering problem via minimum spanning sphere. In *ICALP 2011: Automata, Languages and Programming*, volume 6755 of *LNCS*, pages 773–784. 2011.
20. Z. Drezner. On the rectangular p-center problem. *Naval Research Logistics*, 34(2):229–234, 1987.
21. M. E. Dyer. On a multidimensional search technique and its application to the euclidean one-centre problem. *SIAM Journal on Computing*, 15(3):725–738, 1986.
22. D. Eppstein. Faster construction of planar two-centers. In *Proc. 8th Annual ACM-SIAM Symposium on Discrete algorithms*, SODA '97, pages 131–138, 1997.
23. H. N. Gabow, J. L. Bentley, and R. E. Tarjan. Scaling and related techniques for geometry problems. In *Proc. 16th Annual ACM Symposium on Theory of Computing*, STOC '84, pages 135–143, 1984.
24. J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Information Processing Letters*, 33(4):169–174, 1989.
25. D. P. Huttenlocher, K. Kedem, and M. Sharir. The upper envelope of voronoi surfaces and its applications. In *Proc. 7th Annual Symposium on Computational Geometry*, SCG '91, pages 194–203, 1991.
26. H. Imai, M. Iri, and K. Murota. Voronoi diagrams in the Laguerre geometry and its applications. *SIAM J. Comput.*, 14(1):93–105, 1985.
27. A. Jørgensen, M. Löffler, and J. M. Phillips. Geometric computations on indecisive points. In *WADS 2011: Algorithms and Data Structures*, volume 6844 of *LNCS*, pages 536–547. 2011.
28. M. Korman and T. Tokuyama. Optimal insertion of a segment highway in a city metric. In *COCOON 2008: Computing and Combinatorics*, volume 5092 of *LNCS*, pages 611–620, 2008.
29. D. T. Lee and Y. F. Wu. Geometric complexity of some location problems. *Algorithmica*, 1:193–211, 1986.
30. M. Löffler. *Data Imprecision in Computational Geometry*. PhD thesis, Utrecht University, 2009.
31. N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983.
32. M. Sharir and P. K. Agarwal. *Davenport-Schinzel sequences and their geometric applications*. Cambridge University Press, 1995.
33. K. Sheth, T. Islam, and P. Kopardekar. Analysis of airspace tube structures. In *27th Digital Avionics Systems Conference, IEEE/AIAA*, pages 3.C.2–1–3.C.2–10, 2008.
34. B. Sridhar, S. Grabbe, K. Sheth, and K. Bilimoria. Initial study of tube networks for flexible airspace utilization. In *AIAA Guidance, Navigation, and Control Conference, AIAA-2006-6768*, 2006.
35. A. Yousefi, G. Donohue, and L. Sherry. High volume tube shaped sectors (HTS): A network of high-capacity ribbons connecting congested city pairs. In *23rd Digital Avionics Systems Conference, IEEE/AIAA*, pages –3.1–7 Vol.1, 2004.
36. A. Yousefi and A. N. Zadeh. Dynamic allocation and benefit assessment of NextGen flow corridors. *Transportation Research Part C: Emerging Technologies*, 33(0):297–310, 2013.