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]

[1] Dept. of Appl. Math. and Statistics, State Univ. of NY at Stony Brook, USA
[2] Departamento Matemática Aplicada II, Universidad de Sevilla, Spain
[3] Dept. de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Spain
[4] Dept. of Computer Science, Florida State University, Tallahassee, FL, USA
[5] Departamento de Informática, Universidad de Valladolid, Spain
[6] Esc. de Ingeniería Civil en Informática, Universidad de Valparaíso, Chile
[7] Dept. of Appl. Math., Charles University, Prague, Czech Republic

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, assign to each point a color ("red" or "blue") so that each pair's points are assigned different colors and a function of the radii of the minimum enclosing balls of the red points and the blue points, respectively, is optimized. In particular, we consider the problems of minimizing the maximum and minimizing the sum of the two radii. For each case, minmax and minsum, we consider distances measured in the L_2 and in the L_∞ metrics. Our problems are motivated by a facility location problem in transportation system design, in which we are given origin/destination pairs of points for desired travel, and our goal is to locate an optimal road/flight segment in order to minimize the travel to/from the endpoints of the segment.

1 Introduction

Consider 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 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, and simplified mathematical models have been widely studied in order to investigate basic geometric properties of urban transportation systems [1]. Recently, there has been an interest in facility location problems derived from urban modeling. In many cases we are interested in locating a highway that optimizes some given function that depends on the distance between elements of a given point set (see for example [4,7,9,13]).

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 [14,15,16,17] 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.

We consider a simplified model closely related to the well-known 2-CENTER problem. In the standard 2-CENTER problem, we are given a set of n points representing customers and the goal is to locate two facilities in the plane to minimize the largest Euclidean distance from a customer to its nearest facility. This problem received much attention in recent years; the current best known algorithm is due to Eppstein [12]. The RECTILINEAR 2-CENTER problem, using the L_1- or L_∞-metric, can be solved in linear time [10]. The discrete version was considered by Bespamyatnikh and Segal [6].

In our setting, the set S consists of pairs of points (origin/destination pairs) in the plane. We seek 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. (Refer to Fig. 1, which depicts a scenario in the air traffic setting.) While there are numerous practical considerations when designing optimal transportation corridors, we concentrate on minimizing the distance that traffic must travel outside of the corridor.

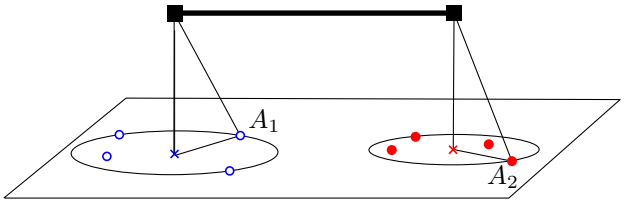


Fig. 1. Schematic of a flow corridor (bold segment) servicing air traffic between blue and red points (airports). Distances between the airports and the endpoints depend on the radii of the disks.

The general optimization problem we study can be formulated as follows:

The 2-CENTER COLOR ASSIGNMENT problem: Given a set S of n pairs of points in the plane and two colors (red and blue), assign different colors to the points of each pair of S, in such a way a function on the size of the minimum enclosing balls of the red and the blue points, respectively, is minimized.

Problems studied. Suppose a color assignment is given and let R and B be the sets of red and blue points, respectively. 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. In each criterion, we study the problem for both the L_∞ and the L_2 metrics. We consider then four variants of the 2-CENTER COLOR ASSIGNMENT problem and they will be referred as: the MINMAX-L_∞ problem, the MINMAX-L_2

	MinMax	MinSum	Pairs of Points 1-Center
L_∞	$O(n)$	$O(n \log^2 n)$	$O(n \log^2 n)$
L_2	$O(n^3 \log^2 n)$ $O(n/\varepsilon^6)$ $(1+\varepsilon)$-approx	$O(n^5 \operatorname{polylog} n)$ $O(n/\varepsilon^6)$ $(1+\varepsilon)$-approx	$O(n^2 \operatorname{polylog} n)$

Table 1. Summary of the running times of the algorithms.

problem, the MinSum-L_∞ problem, and the MinSum-L_2 problem. A natural simplification of these problems is the Pairs of Points 1-Center problem which consists in finding a minimum ball enclosing 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.

It is worth noting that the restriction on the coloring of pairs of points makes our problems rather different from the classic 2-center problem, 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 aim 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 [8]. The objective functions we use are more complex, leading to considerably more complex problems.

Results. We present exact algorithms for all the four variants of the 2-Center Color Assignment problem, with running times summarized in Table 1. In addition, based on our linear-time algorithm for the MinMax-L_∞ problem, we present an $O(n/\varepsilon^6)$-time $(1+\varepsilon)$-approximation 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. In addition, we solve the Pairs of Points L_∞ 1-Center problem and the Pairs of Points L_2 1-Center problem in $O(n \log^2 n)$ and $O(n^2 \operatorname{polylog} n)$ time, respectively. The solution given to these two problems are used in the solutions to the MinSum problems.

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.

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 C_R and C_B have equal radius and that each of the two disks (squares) has one of its vertices coinciding with a corner of H. We consider two fixed vertices of H and anchor C_R and C_B to them. 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. 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. Since we have that H can be found in linear time, there are $O(1)$ combinations of vertices of H to anchor C_R and C_B, and the smallest feasible radius of C_R and C_B for each anchor combination can be computed in linear time, then the next result is obtained.

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

3 The MinMax-L_2 problem

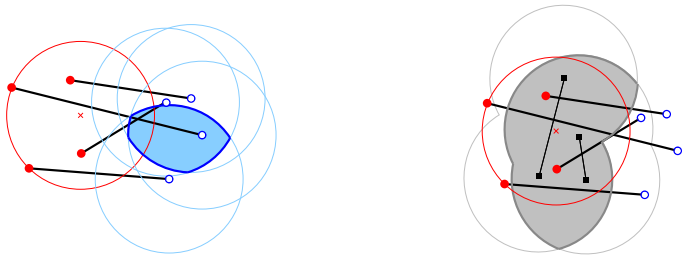
3.1 An exact algorithm

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, that is, it is a solution to the 1-center problem of those points. The overall idea is to perform a binary search on all the candidate radii r for C_R and C_B, testing if there is a feasible solution in which the radius r 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 that we try, we test all disks of radius r that have two points from S on its boundary. Each of those disks will be a candidate for one of the two disks that need to be found. W.l.o.g., we assume that it will be the disk C_R. Once a candidate C_R is fixed, we will test if there exists a feasible second disk C_B with radius at most r. Depending on the latter, the binary search continues in the usual way by increasing or decreasing the value of r. More details on the algorithm follow.

Consider one step of the binary search. In order to decide whether $r^* \leq r$, we need to test $\Theta(n^2)$ disks C_R. These disks correspond to the two disks having radius r through every tuple of points p, q of S (p and q do not necessarily form a pair). For a given C_R, we can decide in $O(n \log n)$ time if C_R and some other disk of radius r form a feasible solution. 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. If each pair of S has only one point in C_R, then 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.
2. If both points of each pair are inside C_R, then $r^* < r$.
3. Otherwise, in the most general case where some (but not all) pairs of points are contained in C_R, we have to assign them colors in order to decide if a blue disk C_B whose radius is at most r exists.



(a) (b)

Fig. 2. (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.

To assign colors to the pairs inside C_R, 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 (Fig. 2(a)). The region I_D is convex, its boundary has linear complexity, and can be computed by using a divide and conquer approach in $O(n \log n)$ time. Let I_{DD} be the intersection of the double disks of radius r centered at 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. 2(b)). Using several geometric properties of both I_D and I_{DD}, we prove the following important lemma in the full version:

Lemma 1. *The intersection $I_D \cap I_{DD}$ can be computed in $O(n \log n)$ time.*

If this intersection is non-empty, then there exist two disks of radius r (one of which is C_R) that form a feasible solution, and thus $r^* \leq r$. Otherwise, we test a new tuple p, q of S or, if all tuples have been considered, we decide that $r^* > r$ and proceed with the binary search. 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^*. Deciding each value of r costs $\Theta(n^2) \cdot O(n \log n) = O(n^3 \log n)$ time, 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 An approximation algorithm

In this section we present a surprisingly simple algorithm that gives an $O(n/\varepsilon^6)$-time $(1+\varepsilon)$-approximation, for any constant ε, $0 < \varepsilon < 1$. Our method is similar to techniques used for approximating the standard k-CENTER problem (e.g. [2]).

First we solve the MINMAX-L_∞ problem in linear time (Section 2) and obtain the squares Q_R and Q_B covering the red and blue points, respectively. Observe that this solution of the MINMAX-L_∞ problem gives a $\sqrt{2}$-approximation. Assume w.l.o.g. that Q_R is larger than Q_B. For simplicity, scale the point set so that Q_R becomes a 1×1 square having circumradius $\sqrt{2}/2$. Let r^* denote the size of optimal disks C_R and C_B of an optimal solution to the MINMAX-L_2 problem. Then we have $1/2 \leq r^* \leq \sqrt{2}/2$.

Next we overlay a square grid on top of the point set (see Fig. 3). Each cell has size $\varepsilon/3 \times \varepsilon/3$. However, we are only interested in grid cells that, together, cover the area where the (unknown) optimal disks C_R and C_B are. To this end it suffices to cover the area of both squares Q_R and Q_B plus a buffer around them of width $\sqrt{2}/2$. In this way, the set of all cells considered, denoted by $\mathcal{C}$, has size $O(1/\varepsilon^2)$.

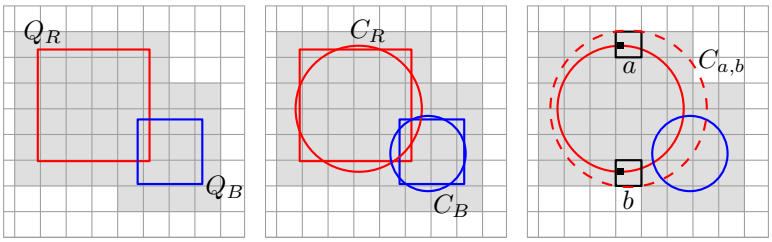


Fig. 3. Left: schematic drawing of two optimal squares and the set of cells $\mathcal{C}$ (shaded). Center: optimal disks C_R and C_B. Right: $C_{a,b}$ is a $(1+\varepsilon)$-approximation of C_R.

The algorithm consists in trying all pairs of disks where each disk is defined by two grid cells. Therefore, we try all the quadruples $\{a, b, c, d\}$ of cells of $\mathcal{C}$, assuming that a and b are diametric points defining the first disk, whereas c and d are diametric points defining the second disk. Furthermore, it is enough to look at cells a, b on the same column (i.e. vertically aligned), and columns c, d on the same column as well. In this way, each quadruple $\{a, b, c, d\}$ gives place to two disks denoted by $C_{a,b}$ and $C_{c,d}$. More precisely, $C_{a,b}$ is defined as the smallest disk that contains cells a and b. Recall that each cell is a $\varepsilon/3 \times \varepsilon/3$ square. $C_{c,d}$ is defined analogously.

We then test each pair $(C_{a,b}, C_{c,d})$ of disks for feasibility, that is, if every pair of points in S contains one of its points in $C_{a,b}$ and the other one in $C_{c,d}$. The feasibility test takes $O(n)$ time. After trying the disks associated with all quadruples of cells, the algorithm returns the feasible pair of disks with smallest maximum radius. Since the algorithm tries $O(1/\varepsilon^3)$ cells for each of the two

candidate disks, the total running time is $O(n/\varepsilon^6)$. It remains only to show that the algorithm computes a $(1+\varepsilon)$-approximation. This is proved in the full version, where we also give an alternate method that leads to a time complexity that is the *sum* of $O(n)$ and $O(\text{poly}(1/\varepsilon))$.

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

4 The 1-center problems for pairs of points

In this section we propose solutions to both the Pairs of Points L_∞ 1-Center problem and the Pairs of Points L_2 1-Center problem.

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

Proof. Consider both the decision and the optimization problem.

Decision Problem: Given a size $d > 0$, does there exist a square of size $2d$ covering at least one point of each pair? It can be solved in $O(n \log n)$ time as follows. For each point p of S, let H_p be the axis-aligned square of size $2d$ centered at p. Given paired points p and q of S, represent the set $H_p \cup H_q$ by the union of at most three rectangles with pairwise disjoint interiors. Let $Q_{p,q}$ 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 $Q_{p,q}$, over all paired points p and q of S, is equal to n. This can be solved in $O(n \log n)$ time [5].

Optimization Problem: 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 $|y(p) - y(q)|$. We proceed now to compute the optimal value for d, which is equal, w.l.o.g., to $|y(p) - y(q)|$ for two points p, q of S. Let $p_1, p_2, \ldots, p_{2n}$ be the points of S sorted by y-coordinate and consider the $2n \times 2n$ matrix M such that:

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

Note that M is a sorted matrix (i.e. every row and every column is sorted) containing all the possible values of d, and we then can apply matrix searching [3] in order to execute the decision procedure $O(\log n)$ times. 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 order of S by y-coordinate. □

Theorem 5. *The* Pairs of Points L_2 1-Center *problem can be solved in $O(n^2 \,\text{polylog}\, n)$ time.*

Proof. We build the planar arrangement induced by all the n bisectors of the pairs of points in S. This arrangement has $O(n^2)$ cells. For each cell we have the

n-point subset $S' \subset S$ including for each pair of S the element that is closest to every point within the cell than the other element. Then we solve the 1-CENTER problem of S' as a potential solution. If the cells of the arrangement are processed in order (i.e. moving only between neighboring cells) then whenever we move from one cell to an adjacent one, one point enters S' and other point exits S'. When it happens the solution to the 1-CENTER problem of S' can be updated in amortized expected $O(\text{polylog}\, n)$ time, by using a suitable dynamic data structure for the DYNAMIC 1-CENTER problem in two dimensions [11]. □

5 The MINSUM-L_∞ problem

Up to symmetry, there are four relative positions of C_R and C_B, as depicted in Fig. 4. 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 and can be found in $O(n \log^2 n)$ time (Theorem 4).

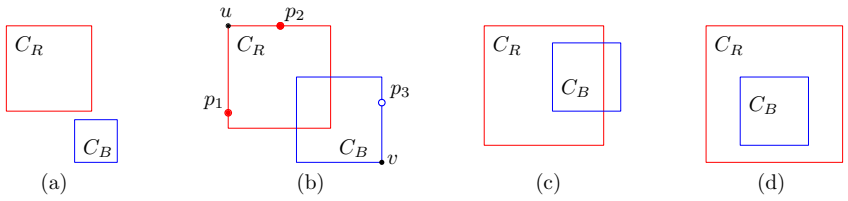


Fig. 4. Relative positions of C_R and C_B.

Let H be the smallest axis-aligned rectangle covering S and let u be its top-left vertex. Let p_1, p_2, and p_3 be the points of S contained on the left, top, and right boundaries of H, respectively. We assume in any of the cases a), b), and c) that p_1 and p_2 belong to the left and top boundaries of C_R, respectively, and also that p_3 and the bottommost point colored blue are on the right and bottom boundaries of C_B, respectively. Observe that from this assumption vertex u is fixed and vertex v is not, where v denotes the bottom-right vertex of C_B.

Algorithm overview: Consider all points of S are *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-right vertex anchored at v. 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. Then, we apply a sweep of 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 black point p. In each event, we color point p blue and its partner red, and considering C'_R fixed, compute the smallest feasible square C'_B. Notice that C'_B, having bottom-right vertex v, covers all points colored blue, and covers, 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. The sweep finishes when the boundary of C'_R hits a point that has been colored red. During the sweep, we keep track of the pair (C'_R, C'_B) minimizing the sum of their radii.

We now proceed to explain how the above sweep can be done in $O(n \log n)$ time. Observe that if both u and v are fixed the sweep can be done in $O(n \log n)$ time. This implies that we can consider only the events where points of S are crossed by the right boundary of C'_R. Indeed, the first time the bottom boundary of C'_R crosses a point p of S, which is in fact the lowest point of S, we color p blue and then from this point forward vertex v must be the bottom-right vertex of box H to ensure that C'_B covers p, and thus v is fixed. Further 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 of them there is no pair of points of S whose two elements are below v. See Fig. 5(a).

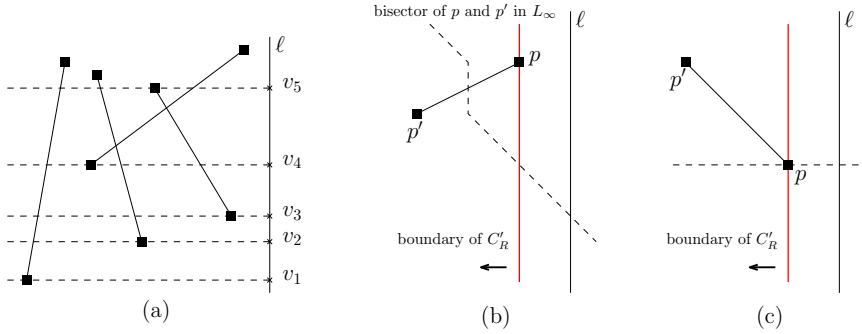


Fig. 5. (a) The possible positions $v_1, \ldots, v_5$ for vertex v. (b) One sweep case. (c) The other sweep case.

Let $v_1, v_2, \ldots, v_k$ denote from bottom to top the possible positions for vertex v. At any moment in the sweep let $r(v_i)$, $i \in [1..k]$, denote the size of the smallest feasible square C'_B having v_i as bottom-right vertex. We design a dynamic efficient data structure (called DS) so that for each sweep event, DS reports in $O(\log n)$ time both v_j and $r(v_j)$ where $j = \arg\min_{j \in [1..k]} r(v_j)$.

At the beginning of the sweep, we first compute $r(v_i)$ for all $i \in [1..k]$. Observe that this computation is a one-dimensional problem. We show in the full version how it can be done in $O(n \log n)$ time. After that, we build DS as follows. DS is a balanced binary tree with k leaves, which are, from left to right, the points $v_1, v_2, \ldots, v_k$. We augment every node z with four values α, β, γ, and ρ so that: α is the minimum $r(\cdot)$ of the leaves descendant of z; β is a reference to a leaf descendant of z minimizing $r(\cdot)$ (i.e. $r(\beta) = \alpha$); γ is a reference to the rightmost leaf descendant of z; and ρ is a point of S attached to z. In this way α and β of the root node determine the best square C'_B. DS can be built in linear time from $r(v_1), r(v_2), \ldots, r(v_k)$. Initially, there is no ρ values attached to any node, they will be attached during the events so that at any moment $r(v_i)$, $i \in [1..k]$,

is equal to the maximum between the initial value of $r(v_i)$ (still stored at leaf v_i) and $\max_\rho\{y(\rho) - y(v_i)\}$ for all points ρ attached at nodes in the path from v_i to the root.

Whenever the right boundary of C'_R crosses a black point p of S, we must update some of the values $r(v_1), r(v_2), \ldots, r(v_k)$ and perform the consequent update of DS. There are two cases to follow according to the relative position of p and its partner, p', which is to the right of p. In Fig. 5(b) and Fig. 5(c) we show these two cases. Observe in the first case (Fig. 5(b)) that we must update $r(v_1), \ldots, r(v_j)$ where $v_1, \ldots, v_j$ are all points among $v_1, v_2, \ldots, v_k$ lying below the bisector of p and p' in L_∞. This must be done since from this point forward the smallest square C'_B with bottom-right vertex v_i, $i \in [1..j]$, must cover point p that is colored blue and furthest from v_i than p'. In the second case (Fig. 5(c)), we must "discard" the points $v_j, v_{j+1}, \ldots, v_k$ lying strictly above the horizontal line through p because point p is now blue. We can discard v_i, $i \in [j..k]$, by considering $r(v_i) = +\infty$. Observe that in each event we always update the elements of an interval of $v_1, v_2, \ldots, v_k$. Such an update can be done as follows. Consider the set Z of nodes of two root-to-leaf paths, the first one connecting the first element of the interval, and the second one connecting the last element. We first attach a point ρ to nodes of Z, and to children nodes of nodes of Z, so that every root-to-leaf path of the tree has a node to which we attach ρ if and only if the leaf belongs to the interval. In the first case we attach the point $\rho = p$, and in the second case we attach the point $\rho = (-\infty, +\infty)$. If we attach ρ to a node z and $\rho(z)$ was attached before in other update operation, then we select between ρ and $\rho(z)$ the point with maximum y-ordinate as the new attached point to z. Once ρ has been attached to those nodes, we perform the following bottom-up update of every node of Z and every child of a node of Z to which we attached ρ. Let z be a node to be updated. Consider that z is not a leaf (the case where z is a leaf is simpler). Let z_1 and z_2 be the left and right children of z, respectively. We update z by considering two independent cases:

Let C_1 (resp. C_2) be the square with bottom-right vertex $\beta(z_1)$ (resp. $\beta(z_2)$) and size $\alpha(z_1)$ (resp. $\alpha(z_2)$). Let C' be the smallest square between C_1 and C_2.

Case (1): *$\rho(z)$ is not attached or $\rho(z) \in C'$.*

If $C' = C_1$ then $\alpha(z) := \alpha(z_1)$ and $\beta(z) := \beta(z_1)$. Otherwise, $\alpha(z) := \alpha(z_2)$ and $\beta(z) := \beta(z_2)$.

Case (2): *$\rho(z)$ is attached and $\rho(z) \notin C'$.*

$\alpha(z) := y(\rho(z)) - y(\gamma(z))$ and $\beta(v) := \gamma(v)$.

The update and query of DS cost $O(\log n)$ time per event, and an optimal solution satisfying case a), b), or c) can then be found in $O(n \log n)$ time. The time complexity is dominated by the $O(n \log^2 n)$-time algorithm to find a solution satisfying case d).

Theorem 6. *The* MinSum-L_∞ *problem can be solved in $O(n \log^2 n)$ time.*

6 The MinSum-L_2 problem

The MinSum-L_2 problem can be solved by considering each possible disk C_R that contains at least one point from each pair. Then for each election of C_R some pairs are colored and the other pairs are not. Then we 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 adapts to the Pairs of Points L_2 1-Center problem, and can thus be solved in $O(n^2 \operatorname{polylog} n)$ time (Theorem 5). This implies an overall $O(n^5 \operatorname{polylog} n)$-time algorithm.

Given the high running time of the algorithm, it is worth noting that almost the same $O(n)$-time approximation of Section 3.2 can be applied to this problem as well. The only difference is the initial constant-factor approximation used. 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, exactly the same approach leads to a $(1+\varepsilon)$-approximation for this problem, running in $O(n/\varepsilon^6)$ time.

Theorem 7. *The* MinSum-L_2 *problem can be solved in* $O(n^5 \operatorname{polylog} n)$ *time. A* $(1+\varepsilon)$*-approximation can be computed in* $O(n/\varepsilon^6)$ *time for any* $\varepsilon \in (0,1)$.

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. On the other hand, it would also be interesting to obtain lower bounds results for the problems studied, and to extend them to 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) and by Metron Aviation (subcontract from NASA Ames). J.M. D.-B. was partially supported by project FEDER MEC MTM2009-08652. P. P.-L. was partially supported by project FEDER MEC MTM2009-08652 and grant FONDECYT 11110069. 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. M. S. was funded by ESF Project No. 10-EuroGIGA-OP-003 GraDR. R.I. S. was funded by the FP7 Marie Curie Actions Individual Fellowship PIEF-GA-2009-251235. 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. B.P. was partially supported by MTM2008-05043.

References

1. 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.
2. P. K. Agarwal and C. M. Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002.
3. P. K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. *ACM Computing Surveys*, 30(4):412–458, 1998.
4. 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.
5. H. Alt and L. Scharf. Computing the depth of an arrangement of axis-aligned rectangles in parallel. In *Abstracts 26th European Workshop on Computational Geometry*, pages 33–36, 2010.
6. S. Bespamyatnikh and M. Segal. Rectilinear static and dynamic discrete 2-center problems. In *Algorithms and Data Structures*, Springer *LNCS*, vol. 1663, pages 276–287, 1999.
7. 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.
8. J. Cardinal and S. Langerman. Min-max-min geometric facility location problems. In *Abstracts 22nd European Workshop on Computational Geometry*, pages 149–152, 2006.
9. J. M. Díaz-Báñez, M. Korman, P. Pérez-Lantero, and I. Ventura. Locating a service facility and a rapid transit line. In *Proc. 14th Spanish Meeting on Computational Geometry*, pages 189–192, 2011.
10. Z. Drezner. On the rectangular p-center problem. *Naval Research Logistics*, 34(2):229–234, 1987.
11. D. Eppstein. Dynamic three-dimensional linear programming. *INFORMS Journal on Computing*, 4(4):360–368, 1992.
12. D. Eppstein. Faster construction of planar two-centers. In *Proc. 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 131–138, Philadelphia, PA, USA, 1997.
13. M. Korman and T. Tokuyama. Optimal insertion of a segment highway in a city metric. In *Proc. 14th International Conference on Computing and Combinatorics*, Springer *LNCS*, vol. 5092, pages 611–620, 2008.
14. K. Sheth, T. Islam, and P. Kopardekar. Analysis of airspace tube structures. In *27th Digital Avionics Systems Conference, IEEE/AIAA*, Oct. 2008.
15. 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*, Keystone, CO, Aug. 2006.
16. 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*, Salt Lake City, UT, 2004.
17. A. Zadeh, A. Yousefi, and A. A. Tafazzoli. Dynamic allocation and benefit assessment of NextGen flow corridors. In *4th International Conference on Research in Air Transportation*, Budapest, Hungary, June 2010.