A Simple Linear Algorithm for Computing Rectilinear 3-Centers*

Michael Hoffmann**

Institute for Theoretical Computer Science
ETH Zürich, CH-8092 Zürich, Switzerland
hoffmann@inf.ethz.ch

Abstract. Rectilinear k-centers of a finite point set $\mathcal{P} \subset \mathbb{R}^2$ are the centers of at most k congruent axis-parallel squares of minimal size whose union covers $\mathcal{P}$. This paper describes a linear time algorithm based on the prune-and-search paradigm to compute rectilinear 3-centers. The algorithm is elementary in the sense that it does not build on any sophisticated data structures or other algorithms, except for linear time median finding. An implementation is publically available as part of the Computational Geometry Algorithms Library (CGAL).

Keywords: geometric optimization, facility location, rectilinear center.

1 Problem Statement

The k-center problem is one of the basic problems in facility location. The aim is to locate a set of k facilities for a given set of demand points, such that for any demand point the nearest facility is as close as possible. In a geometric setting, demand points and facilities are interpreted as points in $\mathbb{R}^d$.

This paper is concerned with the *rectilinear k-center problem*, where the distance between points is measured according to the rectilinear (l_1 or l_∞) metric. Formally, this problem is defined as follows. Consider a finite set $\mathcal{P}$ of points in the plane. For a positive integer k the *k-radius* $\varrho_k(\mathcal{P})$ is the minimal $\varrho \in \mathbb{R}$ such that $\mathcal{P}$ can be covered by the union of k congruent closed axis-parallel squares of side length 2ϱ. The *rectilinear k-center problem* is to compute $\varrho_k(\mathcal{P})$ and a corresponding covering for $\mathcal{P}$ (Fig. 1). Note that the centers of the covering squares correspond to the facility placements mentioned above. If there exists a k-covering with radius ϱ, then k facilities can be placed such that for each demand point there is at least one facility in (rectilinear) distance at most ϱ.

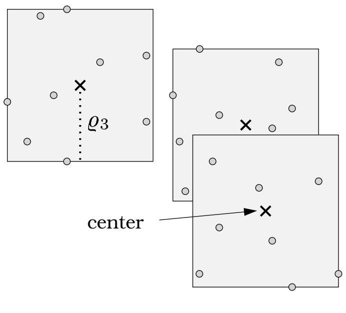


Fig. 1. An instance of the rectilinear 3-center problem.

* A preliminary version of this paper appeared in Proc. 11th Canadian Conference on Computational Geometry (Vancouver, 1999).

** This work is partially supported by the ESPRIT IV LTR Projects No. 21957 (CGAL) and 28155 (GALIA), and by the Swiss Federal Office for Education and Science (CGAL and GALIA).

If k is part of the input, the corresponding decision problem, that is to decide whether a k-covering with some fixed radius ϱ exists, is known to be NP-complete [25]. Moreover, the k-radius cannot be approximated within a factor of less than two, unless $P = NP$ [21]. In contrast to these intractability results, one can solve the problem efficiently for small k.

For $k \leq 3$, the problem can be solved in linear time [30]. More precisely, for $k = 2$ Drezner [8] was the first who gave a linear time algorithm. Rectilinear 4– and 5-centers can be computed in $\mathcal{O}(n \log n)$ time [27, 28] which is optimal in the algebraic computation tree model [30]. These results lead to $\mathcal{O}(n^{k-4} \log n)$ solutions for the general problem. For $k < 90$, this is better than the asymptotically best algorithm known so far [1] which has complexity $\mathcal{O}\left(n^{8\sqrt{k}+10} \log n\right)$. Already in three dimensions, rectilinear 3-centers cannot be computed in linear time: there is a lower bound of $\Omega(n \log n)$ in the algebraic computation tree model [15]. Conversely, there exists an algorithm of matching complexity that works up to dimension five [3].

Related Results

Many variants of the k-center problem have been studied. In the *discrete k-center* problem [2, 4, 17], facilities can be placed at input points only (or at some other finite set of candidate positions). If one wants to point out that there is no such restriction, one speaks about *unrestricted* or *continuous* k-centers. Several papers consider a *weighted* version [23, 26, 20, 19], where each demand point carries a real weight and the distances to the facilities are weighted accordingly. Further variations include the *capacitated k-center* problem [18], where each facility can serve a certain number of demand points only, and an *obstacle* scenario [14], where the centers must not lie inside certain polygonal regions.

Finally, the *Euclidean k-center* problem should be mentioned. The setting is the same as in the rectilinear case, except that distances are measured according to the Euclidean or l_2-metric. That is, in the covering notion used above, axis-parallel squares are replaced by circles. The complexity results for the general case are very similar. If k is part of the input, the Euclidean k-center decision problem is NP-complete [25]. Moreover, the Euclidean k-radius cannot be approximated within a factor of less than 1.822, unless $P = NP$ [11]. For $k = 1$, the problem can be solved in linear time in any fixed dimension [9]. But already the case $k = 2$, that is the complexity of computing Euclidean 2-centers, is not fully resolved yet. A series of papers [16, 29, 10, 7] established the current deterministic bound of $\mathcal{O}\left(n \log^2 n \log^2 \log n\right)$ and a randomized algorithm that runs in $\mathcal{O}\left(n \log^2 n\right)$ expected time.

Outline

For the remainder of this paper, the focus will be on the case $k = 3$, that is on computing rectilinear 3-centers. As noted above, there already exists a linear time algorithm [30] for this case that uses LP-type machinery [22]. But the 3-center problem as an LP-type problem has a rather high combinatorial dimension (roughly 40). Apart from the fact that implementing this algorithm is a non-trivial task, it is unlikely to perform well in practice: there is no algorithm known which solves LP-type problems in time polynomial in their combinatorial dimension.

In this paper, geometric arguments are used to devise an algorithm based on the *prune-and-search* (or *decimation*) paradigm. This technique has been introduced by Megiddo [24] to show that the solution to a linear program with n constraints in d variables can be computed in $\mathcal{O}(n)$ time for any fixed d. At each step, a constant fraction of the input is discarded and not considered anymore in the following. If any single step can be done in linear time, the runtime $T(n)$ in terms of the input size n can be recursively bounded by $T(n) \leq cn + T(\alpha n) = \mathcal{O}(n)$, for constants $c \in \mathbb{R}$ and $\alpha \in (0, 1)$.

The technique is first applied to compute one-dimensional 3-centers in Section 2. Then, Section 3 explains how the ideas can be generalized to solve the planar rectilinear 3-center problem. This leads to a linear time algorithm with good performance in practice, as the experimental results presented in Section 4 confirm.

Throughout the paper some terminology is used that is common to many general optimization problems. For example, any set of three (congruent axis-parallel) squares whose union covers the input set $\mathcal{P}$ is considered to be a *solution* to the problem. In our context, a solution is also referred to as *covering* for $\mathcal{P}$. In an *optimal solution*, the size of the squares is minimal among all solutions. Often, the term *optimal solution* is also used to denote a solution that is optimal among a certain type of solutions, that is for a subset of the solution space. To distinguish this partial optimum from the the global optimum, the latter is called *overall optimum*.

2 One-Dimensional 3-Centers

To illustrate the main ideas, let us first look at the one-dimensional 3-center problem. We are given a set $\mathcal{P}$ of n real numbers and have to find three closed congruent intervals of minimal size whose union covers $\mathcal{P}$ (Fig. 2). Observe that Euclidean and rectilinear metric are the same in one dimension. Hence, the term 3-center is used without further qualification. The result developed below is not too interesting in itself, but should be considered a proof of concept and warmup for the two-dimensional problem that is discussed in Section 3.

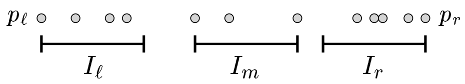


Fig. 2. The one-dimensional 3-center problem.

In any arbitrary covering, the intervals can be ordered from left to right. Denote by I_ℓ the leftmost interval, by I_r the rightmost interval, and by I_m the interval in between. (If some intervals happen to be identical, distribute the corresponding labels among them arbitrarily.) Consider an optimal solution (I_ℓ, I_m, I_r) for a specific instance.

Proposition 1. *We can assume that the left endpoint of I_ℓ is the smallest value p_ℓ of $\mathcal{P}$.*

Proof. Obviously, p_ℓ cannot lie to the left of I_ℓ, since then it would not be covered by any of the intervals. On the other hand, there is no value from $\mathcal{P}$ to the left of p_ℓ. Hence, we can replace any interval among I_ℓ, I_m and I_r which contains p_ℓ by the interval of the same size whose left endpoint is p_ℓ. The obtained solution is at least as good as the original solution (I_ℓ, I_m, I_r). □

Proposition 2. *We can assume that the right endpoint of I_r is the largest value p_r of $\mathcal{P}$.*

Proof. Argue in the same way as in Proposition 1. Both arguments do not conflict with each other: If any of the intervals contains both p_ℓ and p_r, it can be replaced by the interval $[p_\ell, p_r]$ to obtain a solution which is at least as good as the original solution (I_ℓ, I_m, I_r). □

Since the positions of I_ℓ and I_r are determined by p_ℓ and p_r only, it is straightforward to solve the so called 3*-center decision problem*. In this problem one is given a certain radius ϱ and has to tell, whether or not a 3-covering of radius ϱ exists. If such a covering exists, call the corresponding radius *feasible*, otherwise the radius is said to be *infeasible*.

For a value $p \in \mathcal{P}$, define $\gamma(p) := \min\{p - p_\ell, p_r - p\}$. Note that p is covered by I_ℓ or by I_r in any optimal covering with radius ϱ, if and only if $\gamma(p) \leq 2\varrho$. Hence, the set of numbers that have to be covered by I_m in such a covering is $\Gamma(\varrho) := \{p \in \mathcal{P} \mid \gamma(p) > 2\varrho\}$.

Lemma 3. *The one-dimensional* 3*-center decision problem can be solved in linear time.*

Proof. For any value $p \in \mathcal{P}$, computing $\gamma(p)$ takes constant time. Thus, for a given radius ϱ, the set $\Gamma(\varrho)$ as well as its minimum and maximum element can be computed in linear time. A 3-covering with radius ϱ exists, if and only if $\Gamma(\varrho) = \emptyset$ or $\max \Gamma(\varrho) - \min \Gamma(\varrho) \leq 2\varrho$. □

The algorithm from Lemma 3 can be used to perform a binary search on the multiset $\mathcal{C} := \{\frac{1}{2}\gamma(p) | p \in \mathcal{P}\}$ of candidate radii. As such, this binary search takes $\mathcal{O}(n \log n)$ time. But by pruning away a constant fraction of the input values at any step, the runtime can be improved to linear.

Proposition 4. *If the current test-radius ϱ is infeasible, all but two values from $\mathcal{P} \setminus \Gamma(\varrho)$ can be discarded.*

Proof. The main step of the algorithm consists of computing the set $\Gamma(\varrho)$. Since ϱ is infeasible, the binary search will henceforth test radii greater than ϱ only. For any $\sigma > \varrho$, it is $\Gamma(\sigma) \subseteq \Gamma(\varrho)$. Thus, elements from $\mathcal{P} \setminus \Gamma(\varrho)$ need not be considered anymore when computing $\Gamma(\sigma)$. Only the at most two extreme values p_ℓ and p_r have to be kept, since they determine the position of I_ℓ and I_r. □

Proposition 5. *If the current test-radius ϱ is feasible, all but two values from $\Gamma(\varrho)$ can be discarded.*

Proof. Similar to Proposition 4, it is $\Gamma(\sigma) \supseteq \Gamma(\varrho)$ for any future test-radius $\sigma \leq \varrho$. The main step of the algorithm described in Lemma 3 consists of computing the two extreme values of $\Gamma(\sigma)$. Observe that the two extreme values of $\Gamma(\varrho)$ are the only values from $\Gamma(\varrho)$ that can possibly pop up as extreme values of $\Gamma(\sigma)$ as well. Thus, one can discard the numbers from the set $\Gamma(\varrho)$, except for its at most two extreme values. □

Before describing the algorithm, let us argue why the above mentioned multiset $\mathcal{C}$ of candidate radii is sufficient. Consider an optimal covering (I_ℓ, I_m, I_r) for some input set $\mathcal{P}$. Obviously, at least one of the three intervals is of the form $[p, q]$ with $p, q \in \mathcal{P}$. Otherwise, one could simultaneously shrink all intervals by some small but positive ϵ without changing the set of values from $\mathcal{P}$ that are covered. But then the original intervals would not form an optimal solution. If one of I_ℓ or I_r is bounded by two values from $\mathcal{P}$, the optimal radius is in $\mathcal{C}$. But what if only I_m has this property (as in Fig. 2)? Indeed, in this case the optimal radius is not in the candidate set, and the radius computed by binary search on $\mathcal{C}$ might be too large. But a simple post-processing step provides the correct answer for this case as well.

Proposition 6. *Knowing the smallest feasible radius from $\mathcal{C}$, the optimal radius can be computed in linear time.*

Proof. Let σ be the smallest feasible radius from $\mathcal{C}$ and let τ be the predecessor of σ (largest element smaller than σ) in $\mathcal{C}$. Obviously, τ can be computed in linear time. Since τ is infeasible, the two extreme values q_ℓ and q_r of $\Gamma(\tau)$ are more than 2τ apart from each other. Again, q_ℓ and q_r can be computed in linear time. The interval $I_m = [q_\ell, q_r]$ defines a 3-covering for $\mathcal{P}$ which is optimal, if $q_r - q_\ell < 2\sigma$. The set of values from $\mathcal{P}$ covered by $I_\ell \cup I_r$ is identical for all radii in the open interval (τ, σ). Hence, $I_m = [q_\ell, q_r]$, that is radius $\frac{1}{2}(q_r - q_\ell)$, is the only other radius that has to be taken into account. □

Algorithm 1.
Input: A set $\mathcal{P} \subset \mathbb{R}$ of n real numbers.
Output: A 3-covering for $\mathcal{P}$.

(1) While $|\mathcal{P}| > 5$, do
(2) Compute the lower median μ of the multiset $\mathcal{C} := \{\frac{1}{2}\gamma(p) \mid p \in \mathcal{P}\}$.
Let π be a value from $\mathcal{P}$ for which $\gamma(\pi) = 2\mu$.
(3) Test feasibility of μ. Let Λ be the set of extreme values of $\Gamma(\mu)$. ($0 \leq |\Lambda| \leq 2$)
(4) If μ is feasible, let $\mathcal{P} \leftarrow \{p \in \mathcal{P} \mid \gamma(p) < 2\mu\} \cup \{\pi\} \cup \Lambda$.
(5) If μ is infeasible, let $\mathcal{P} \leftarrow \Gamma(\mu) \cup \{p_\ell, p_r\}$.
(6) Solve the problem brute-force.

The *rank-k element* of a multiset M, for $1 \le k \le |M|$, is an element $\mu_k \in M$ for which $|\{m \in M \mid m < \mu_k\}| < k$ and $|\{m \in M \mid m > \mu_k\}| \le |M| - k$. Observe that μ_k is unique for every $k \in \{1, \ldots, |M|\}$, but M might contain several copies of μ_k. The *lower median* of M is its rank-$\lceil \frac{1}{2}|M| \rceil$ element.

Theorem 7. *The one-dimensional 3-center problem can be solved in linear time.*

Proof. Let $n := |\mathcal{P}|$. If μ is feasible, less than $\lceil \frac{n}{2} \rceil + 3$ values remain after Step (4). Similarly, if μ happens to be infeasible, at most $n - \lceil \frac{n}{2} \rceil + 2$ values remain after Step (5). Altogether, at most $\lceil \frac{n+4}{2} \rceil$ values survive a single iteration.

For $n \ge 6$, it is $\lceil \frac{n+4}{2} \rceil < n$, and at least one element is discarded from $\mathcal{P}$. Thus, Algorithm 1 terminates. Each iteration of the loop takes linear time: Step (2) using the algorithm of Blum et al. [5], and Step (3) using Lemma 3. Hence, the runtime $T(n)$ can be bounded by $T(n) \le cn + T(\lceil \frac{n+4}{2} \rceil) < 2cn + 5c \lceil \log_2 n \rceil + d$, for some positive constants $c, d \in \mathbb{R}$. □

3 Planar Rectilinear 3-Centers

This section explains how the ideas and concepts presented in the previous section can be applied to solve the two-dimensional rectilinear 3-center problem. Let $\mathcal{P}$ be a set of n points in the plane. Recall that the goal is to cover $\mathcal{P}$ by the union of three congruent closed axis-parallel squares of minimal size. Instead of two extreme points as in the one-dimensional case, there is now a collection of (up to) four extreme points that define the axis-parallel bounding box $\mathcal{B}(\mathcal{P}) = [x_l, x_r] \times [y_b, y_t]$ of $\mathcal{P}$. Denote by $w(\mathcal{P}) := \min\{x_r - x_l, y_t - y_b\}$ the length of the smallest side of $\mathcal{B}(\mathcal{P})$. A first goal is to solve the decision problem for a specific radius ϱ, using a procedure similar to the one described in Lemma 3.

Proposition 8. *We can assume $2\varrho < w(\mathcal{P})$.*

Proof. If $2\varrho \ge w(\mathcal{P})$, we are back to the one-dimensional problem from Section 2. □

Denote by $s_0(\varrho)$, $s_1(\varrho)$ and $s_2(\varrho)$ the three squares of an optimal 3-covering for $\mathcal{P}$ with radius ϱ. The following statements are two-dimensional versions of Proposition 1 and 2.

Proposition 9. *[30] We can assume $s_i(\varrho) \subseteq \mathcal{B}(\mathcal{P})$ for all $i \in \{0, 1, 2\}$.*

Proof. If, for example, $s_0(\varrho)$ is not contained in $\mathcal{B}(\mathcal{P})$, it can be continuously translated towards $\mathcal{B}(\mathcal{P})$ in x-direction, until the point where it is contained in $\mathcal{B}(\mathcal{P})$. (Recall that $2\varrho < w(\mathcal{P})$ according to Proposition 8.) Denote the translated square by $s_0'(\varrho)$. Since there is no point from $\mathcal{P}$ outside $\mathcal{B}(\mathcal{P})$, it is $\mathcal{P} \cap s_0(\varrho) \subseteq \mathcal{P} \cap s_0'(\varrho)$. That is, the covering $(s_0'(\varrho), s_1(\varrho), s_2(\varrho))$ is at least as good as the original covering $(s_0(\varrho), s_1(\varrho), s_2(\varrho))$. A similar translation in y-direction yields the claimed result. □

Proposition 10. *[30] We can assume that one of the squares $s_0(\varrho)$, $s_1(\varrho)$, or $s_2(\varrho)$ is placed at a corner of $\mathcal{B}(\mathcal{P})$. That is, one of the corners of that square is also a corner of $\mathcal{B}(\mathcal{P})$.*

Proof. Each of the four line segments bounding $\mathcal{B}(\mathcal{P})$ contains at least one point from $\mathcal{P}$. Together with Proposition 9 this means that each of these segments is touched by at least one of the covering squares. There are four segments, but only three covering squares. Thus, one covering square must touch two sides of $\mathcal{B}(\mathcal{P})$. Since $2\varrho < w(\mathcal{P})$, no covering square can touch two opposite sides of $\mathcal{B}(\mathcal{P})$. □

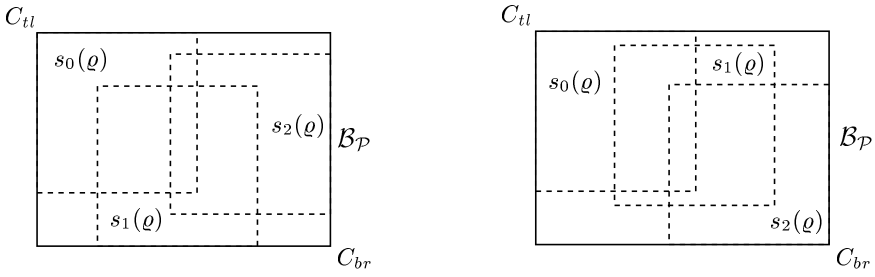
From now on it is assumed that $s_0(\varrho)$ is placed at the top-left corner C_{tl} of $\mathcal{B}(\mathcal{P})$. To really compute an optimal 3-covering for $\mathcal{P}$, the other three symmetric cases, where $s_0(\varrho)$ is placed at one of the other corners, have to be investigated as well. The best covering found in any of these four subcases is the overall optimal covering.

Proposition 11. *[30] Assume there is an optimal* 3*-covering with radius* ϱ *for* $\mathcal{P}$ *where the square* $s_0(\varrho)$ *is placed at the top-left corner* C_{tl} *of* $\mathcal{B}(\mathcal{P})$*. Then there exists an optimal* 3*-covering for* $\mathcal{P}$ *that is of one of the following two types.*

1. *Either* $s_1(\varrho)$ *touches the bottom side and* $s_2(\varrho)$ *touches the right side of* $\mathcal{B}(\mathcal{P})$ *(Fig. 3(a)),*
2. *or* $s_2(\varrho)$ *is placed at the bottom-right corner* C_{br} *of* $\mathcal{B}(\mathcal{P})$ *(Fig. 3(b)).*

Proof. Since $2\varrho < w(\mathcal{P})$, a bottommost point p_b and a rightmost point p_r of $\mathcal{P}$ are not in $s_0(\varrho)$. Let $U := \mathcal{P} \setminus s_0(\varrho)$ denote the set of points not covered by $s_0(\varrho)$. Assume $w(U) \leq 2\varrho$, that is, for example the x-dimension of $\mathcal{B}(U)$ does not exceed 2ϱ. Then both $s_1(\varrho)$ and $s_2(\varrho)$ can be placed such that they touch the right side of $\mathcal{B}(\mathcal{P})$. Moreover, the bottommost of the two squares, say $s_1(\varrho)$, has to cover p_b, and thus, touches the bottom side of $\mathcal{B}(\mathcal{P})$ as well. Overall this yields a configuration of type 2.

On the other hand, if $w(U) > 2\varrho$, none of $s_1(\varrho)$ and $s_2(\varrho)$ can intersect two opposite sides of $\mathcal{B}(U)$. $\mathcal{B}(U)$ has are four extreme sides, and each side has to be touched by at least one covering square. Since there are only two covering squares, they must be placed at two opposite corners of $\mathcal{B}(U)$. If they are placed at the top-left and at the bottom-right corner of $\mathcal{B}(U)$, the configuration is again of type 2. Otherwise, it is a type 1 configuration. □



(a) Type 1. (b) Type 2.

Fig. 3. Two possible placements for $s_1(\varrho)$ and $s_2(\varrho)$.

Proposition 12. *The two types of optimal coverings mentioned in Proposition 11 can be computed in linear time for any given radius* ϱ*,* $0 \leq \varrho < \frac{1}{2}w(\mathcal{P})$*.*

Proof. Denote by $U := \mathcal{P} \setminus s_0(\varrho)$ the set of points not covered by $s_0(\varrho)$. Compute $\mathcal{B}(U) = [x_\ell, x_r] \times [y_b, y_t]$ in linear time. Let $x := \min\{x_\ell, x_r - 2\varrho\}$ and $y := \max\{y_t, y_b + 2\varrho\}$. The position of $s_0(\varrho)$ is determined by ϱ alone. The proof of Proposition 11 shows that there are essentially two possible placements for the squares $s_1(\varrho)$ and $s_2(\varrho)$. For a type 1 configuration, it is $s_1(\varrho) = [x, x + 2\varrho] \times [y_b, y_b + 2\varrho]$ and $s_2(\varrho) = [x_r - 2\varrho, x_r] \times [y - 2\varrho, y]$. For a type 2 configuration, it is $s_1(\varrho) = [x, x + 2\varrho] \times [y - 2\varrho, y]$ and $s_2(\varrho) = [x_r - 2\varrho, x_r] \times [y_b, y_b + 2\varrho]$. □

The two placements of the covering squares mentioned in Proposition 12 will be referred to as *canonical placements* for the given radius ϱ. They depend on ϱ only and on which corner of $\mathcal{B}(\mathcal{P})$ the square $s_0(\varrho)$ is placed at.

Lemma 13. *The rectilinear* 3*-center decision problem for a set* $\mathcal{P} \subset \mathbb{R}^2$ *of* n *points can be solved in* $\mathcal{O}(n)$ *time for any given radius* ϱ*,* $0 \leq \varrho < \frac{1}{2}w(\mathcal{P})$*.*

Proof. For each of the four possible choices to place $s_0(\varrho)$, compute the corresponding canonical placements for ϱ. According to Proposition 12 this can be done in linear time. Checking whether one of these placements is a covering for $\mathcal{P}$ can again be done in linear time. If no covering is among them, radius ϱ is infeasible according to Propositions 10 and 11. □

3.1 Type Two Coverings

The next step is to look at the corresponding optimization problem of finding the optimal radius $\varrho_3(\mathcal{P})$. It turns out that the set of type 2 configurations is handled easily. Its structure is very similar to the one-dimensional problem, since the position of two squares, s_0 and s_2, is independent from the current test-radius ϱ.

Lemma 14. *An optimal rectilinear 3-covering of type 2 (cf. Proposition 11 and Fig. 3(b)) can be computed in linear time.*

Proof. For two points $p, q \in \mathbb{R}^2$ with Cartesian coordinates $p = (p_x, p_y)$ and $q = (q_x, q_y)$ define the rectilinear distance $d_\infty(p, q) := \max\{|p_x - q_x|, |p_y - q_y|\}$. For a point $p \in \mathbb{R}^2$, let $\gamma(p) := \min\{d_\infty(p, C_{tl}), d_\infty(p, C_{br})\}$. Using this definition of γ, apply Algorithm 1 and the machinery from Section 2. The only difference is that whenever we had to deal with two extreme values in the one-dimensional problem, we have to take into account four extreme values now (maximum and minimum in both x– and y-direction). For example, the adaptation of Proposition 5 reads: ... all but four points from $\Gamma(\varrho)$ can be discarded. Repeat the argument for the other three possible placements of $s_0(\varrho)$. □

3.2 Type One Coverings

Within this section, the focus is put on the first type of coverings from Proposition 11 as shown in Fig. 3(a). The terms used, like, e.g., feasibility, have to be interpreted as with respect to this particular type of coverings. The situation is slightly more complicated compared to type 2 coverings, since only the position of one square, s_0, is independent from the current test-radius ϱ.

The idea is to perform again a (sort of) binary search on the set $\mathcal{C} := \{\frac{1}{2}d_\infty(p, C_{tl}) \mid p \in \mathcal{P}\}$ of candidate radii. Consider the canonical placement of type 1 for some test-radius ϱ. Denote by $R_i(\varrho) := s_i(\varrho) \setminus (s_{(i+1) \bmod 3}(\varrho) \cup s_{(i+2) \bmod 3}(\varrho))$, for $i \in \{0, 1, 2\}$, the planar region covered by $s_i(\varrho)$ but not by any of the other two covering squares. Let $F(\varrho) := s_2(\varrho) \cap s_3(\varrho)$ (see Fig. 4). The points from $\mathcal{P}$ in $F(\varrho)$ are called *free*, while the points in $\mathcal{P} \cap R_i$ are said to be *restricted* to $s_i(\varrho)$. In order to achieve a linear time algorithm, a constant fraction of the input points should be discarded after each step of the binary search. The two Propositions below are the two-dimensional counterparts of Proposition 4 and 5 from Section 2.

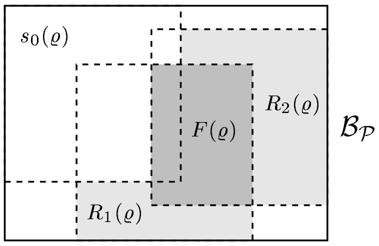


Fig. 4. Free and restricted points.

Proposition 15. *If ϱ is infeasible, all free points and all but two points from $\mathcal{P} \cap s_0(\varrho)$ can be discarded.*

Proof. Since ϱ is infeasible, the binary search will test radii greater than ϱ only from now on. For $\sigma > \varrho$, it is $s_0(\sigma) \supseteq s_0(\varrho)$ and $F(\sigma) \supseteq F(\varrho)$. Only a topmost and a leftmost point from $\mathcal{P}$ in $s_0(\varrho)$ have to be kept, since they define $\mathcal{B}(\mathcal{P})$ and thus the placement of s_0. □

Proposition 16. *If ϱ is feasible, all but eight points from $R_1(\varrho) \cup R_2(\varrho)$ can be discarded.*

Proof. Since ϱ is feasible, the binary search will not test radii greater than ϱ from now on. For any feasible radius $\sigma \leq \varrho$, it is $R_1(\sigma) \supseteq R_1(\varrho)$ and $R_2(\sigma) \supseteq R_2(\varrho)$. Since a point set P is covered by a square s if and only if $\mathcal{B}(P) \subseteq s$, it is sufficient to keep the four extreme points each of $R_1(\varrho)$ and $R_2(\varrho)$. □

As shown in Proposition 6 for the one-dimensional case, it is indeed sufficient to compute the optimal radius from the candidate set $\mathcal{C} = \{\frac{1}{2}d_\infty(p, C_{tl}) \mid p \in \mathcal{P}\}$.

Proposition 17. *Knowing the smallest feasible radius from $\mathcal{C}$, the optimal (type 1) radius can be computed in linear time.*

Proof. Let σ be the smallest feasible radius from $\mathcal{C}$ and let τ be the predecessor of σ in $\mathcal{C}$. Obviously, τ can be computed in linear time. Denote by $U := \mathcal{P} \setminus s_0(\tau)$ the set of points not covered by $s_0(\tau)$. Since τ is infeasible, U cannot be covered by two squares of radius τ. Let C_{bl} denote the bottom-left corner of $\mathcal{B}(U)$, and let C_{tr} denote the top-right corner of $\mathcal{B}(U)$. In a type 1 covering, s_1 has to cover C_{bl}, and s_2 has to cover C_{tr}. For a point $p \in U$ define $\delta(p) := \min\{d_\infty(p, C_{bl}), d_\infty(p, C_{tr})\}$. The minimal radius φ for which $s_1(\varphi)$ and $s_2(\varphi)$ together cover U, is then determined by $2\varphi = \max_{p \in U} \delta(p)$. If $\mathcal{B}(U)$ is known, $\delta(p)$ can be computed in constant time for any $p \in U$. Thus, φ can be computed in linear time. The set of points from $\mathcal{P}$ covered by $s_0(\varrho)$ is identical for all radii in the open interval (τ, σ). Hence, radius φ is the only other radius that has to be taken into account, and it is optimal, if and only if $\varphi \leq \sigma$. □

Below follows the description of the algorithm to compute an optimal type 1 covering. It has a parameter k, that encodes which element of the candidate set is chosen as test-radius to guide the search process. The choice of k is discussed below.

Algorithm 2.
Input: A set $\mathcal{P} \subset \mathbb{R}^2$ of n points, a number $k \in \{1, \ldots, n\}$.
Output: A rectilinear 3-covering for $\mathcal{P}$.

(1) While $|\mathcal{P}| > 27$, do
(2) Compute the rank-k element μ of the multiset $\mathcal{C} := \{\frac{1}{2}\gamma(p) \mid p \in \mathcal{P}\}$.
(3) Compute the canonical placement $(s_0(\mu), s_1(\mu), s_2(\mu))$ for $\mathcal{P}$.
(4) If $(s_0(\mu), s_1(\mu), s_2(\mu))$ do not cover $\mathcal{P}$, then
(5) discard all points in $s_0(\mu) \cup F(\mu)$ from $\mathcal{P}$,
(6) but keep a topmost point and a leftmost point from $\mathcal{P} \cap s_0(\mu)$.
(7) Else if $|R_1(\mu) \cup R_2(\mu)| \geq k + 6$,
(8) discard all points in $R_1(\mu) \cup R_2(\mu)$ from $\mathcal{P}$,
(9) but keep at most four extreme points each of $R_1(\mu)$ and $R_2(\mu)$.
(10) Else repeat Step (3)–(9) where μ is replaced by some magic radius μ'.
(11) Compute the optimal covering for $\mathcal{P}$ brute-force.

Proposition 18. *One iteration of Algorithm 2 (Step (2)–(9)) can be handled in $\mathcal{O}(|\mathcal{P}|)$ time.*

Proof. Every single step can be handled in linear time: Step (2) using the algorithm of Blum et al. [5] and Step (3) as described in Proposition 12. As soon as the canonical placement is known, it takes one iteration over $\mathcal{P}$ to compute the intersection of $\mathcal{P}$ with each of the sets $R_1(\mu)$, $R_2(\mu)$, $s_0(\mu)$, and $F(\mu)$, along with the extreme points of these sets. □

So how should the parameter k be chosen? Ideally, the cardinalities of the sets $s_0(\mu) \cup F(\mu)$ and $R_1(\mu) \cup R_2(\mu)$ are about the same. Then roughly k points can be discarded, regardless of whether or not the canonical placement for μ turns out to be a covering. However, there is no easy way to tell in advance how many points from $\mathcal{P}$ end up in $R_1(\mu)$, $R_2(\mu)$, or $F(\mu)$ for a specific value of k: these numbers can range anywhere between zero and $n - k$. If radius μ is infeasible, at least $k - 2$ points from $s_0(\mu)$ can be discarded. But if μ is feasible, something has to be done in case there are too few points from $\mathcal{P}$ in $R_1(\mu) \cup R_2(\mu)$. This is where the magic value mentioned in Step (10) comes into play.

Lemma 19. *There is a set $C \subset \mathcal{C}$ of four radii, such that after testing these radii for feasibility*

- *either the optimal (type 1) solution can be computed directly in $\mathcal{O}(|\mathcal{P}|)$ time,*
- *or at least $\lfloor \frac{1}{6}(|\mathcal{P}| - 22) \rfloor$ points from $\mathcal{P}$ can be discarded in Step (5) or Step (8) of Algorithm 2.*

Moreover, such a set C can be computed in $\mathcal{O}(|\mathcal{P}|)$ time.

Proof. Let $n := |\mathcal{P}|$. For some radius $\varrho \in \mathbb{R}$, let $f(\varrho) := |\mathcal{P} \cap (F(\varrho) \setminus s_0(\varrho))|$ be the number of free points not covered by $s_0(\varrho)$. Similarly denote by $r(\varrho) := |\mathcal{P} \cap (R_1(\varrho) \cup R_2(\varrho))|$ the number of points restricted to one of $s_1(\varrho)$ or $s_2(\varrho)$. Assume Algorithm 2 is run with some parameter k. If the canonical placement with respect to μ does not provide a covering for $\mathcal{P}$, there are at least $k-2$ points discarded in Step (5). If μ is feasible and $r(\mu) \geq k+6$, at least $k-2$ points are discarded in Step (8). It remains to consider the case that the test-radius μ is feasible and $r(\mu) < k+6$.

For degenerate point sets it can happen that many points from $\mathcal{P}$ lie on the boundary of $s_0(\mu)$, that is there can be many points in $\mathcal{P}$ for which $\gamma(\cdot) = 2\mu$. As a consequence, it may occur that $r(\mu) + f(\mu) < n - k$. If this is the case, the iteration is repeated with the predecessor $\tilde{\mu}$ of μ in $\mathcal{C}$. If $\tilde{\mu}$ does not exist or happens to be infeasible, the optimal solution can be computed as in Proposition 17 and the algorithm terminates. Otherwise, $\tilde{\mu}$ is feasible and $r(\tilde{\mu}) + f(\tilde{\mu}) \geq n - k$. If $r(\tilde{\mu}) \geq k+6$, at least $k-2$ points are discarded in Step (8). Thus, assume that $r(\tilde{\mu}) < k+6$. In this degenerate case where $r(\mu) + f(\mu) < n - k$, replace μ by $\tilde{\mu}$ for the remainder of the proof.

Consider the planar region $F^\circ := F(\mu) \setminus s_0(\mu)$, and recall that $|\mathcal{P} \cap F^\circ| = f(\mu)$. Let l_r be the rightmost vertical line such that there are at least $\alpha f(\mu)$ points in the closed halfplane to the right of l_r, for some parameter $\alpha \in (0, 1)$. Note that this also implies that there are at least $(1-\alpha)f(\mu)$ points in the closed halfplane to the left of l_r. Computing l_r is once again an instance of the standard selection problem and can thus be done in linear time. Define similar lines for the other three directions, and denote with B the (possibly degenerate) rectangle described by these four lines (Fig. 5(a)). Obviously, it is $B \subseteq F(\mu)$. (But not necessarily $B \subseteq F^\circ$.)

Let μ'' be the smallest radius from $\mathcal{C}$ such that $F(\mu'')$ contains B, and let μ' be its predecessor in $\mathcal{C}$ (Fig. 5(b)). Recall that B depends on the original test-radius μ only. For the purpose of finding μ'', rectangle B is a constant object. That is, it can be tested in constant time whether or not a square of some canonical placement contains B. Computing μ'' (and μ') is thus once again an instance of the standard selection problem and can be done in linear time. Re-iterate Algorithm 2 first with μ'' and then (possibly) with μ'.

Observe that $f(\mu'') \geq f(\mu) - 4\alpha f(\mu) = (1-4\alpha)f(\mu)$, since $B \subseteq F(\mu'')$. That is, if μ'' is infeasible, there are at least $(1-4\alpha)f(\mu)$ points discarded in Step (5). On the other hand, if μ'' is feasible and μ' is infeasible or does not exist, one can directly compute the optimal solution as in Proposition 17. Finally, consider the case that μ' is feasible. Since there is at least one side of B that is not contained in $F(\mu')$, say the right side, all points from $\mathcal{P} \cap F(\mu)$ to the right of (or on) this side are contained in exactly one of $R_1(\mu')$ or $R_2(\mu')$, say $R_2(\mu')$. By definition of B, the number of such points is at least $\alpha f(\mu)$, and these points cannot be extreme points of $R_1(\mu')$. Moreover, except for at most four more extreme points of $R_1(\mu')$, also the $r(\mu)$ points from $\mathcal{P}$ in $R_1(\mu) \cup R_2(\mu)$ can be discarded. Thus, at least $r(\mu') - 4 \geq \alpha f(\mu) - 4 + \max\{0, r(\mu) - 4\}$ points from $\mathcal{P}$ are discarded in Step (8).

The optimal choices for the two parameters k and α can be determined by looking at the number of points discarded in each of the different subcases above. For α, one has to maximize $\min\{\alpha f(\mu) - 4 + \max\{0, r(\mu) - 4\}, (1-4\alpha)f(\mu)\}$ which yields $\alpha = \frac{4 + f(\mu) - \max\{0, r(\mu)-4\}}{5f(\mu)}$. For k, the value is defined by $\max\min\left\{k-2, \frac{1}{5}\left(f(\mu) - 16 + 4\max\{0, r(\mu)-4\}\right)\right\}$. Recall that $r(\mu) + f(\mu) \geq n - k$. Since there is no non-trivial lower bound for $r(\mu)$, solving for k yields $k = \frac{n-10}{6}$. The set $C := \{\mu, \tilde{\mu}, \mu', \mu''\}$ has the claimed properties. □

Lemma 20. *An optimal rectilinear 3-covering of type 1 (cf. Proposition 11 and Fig. 3(a)) can be computed in linear time.*

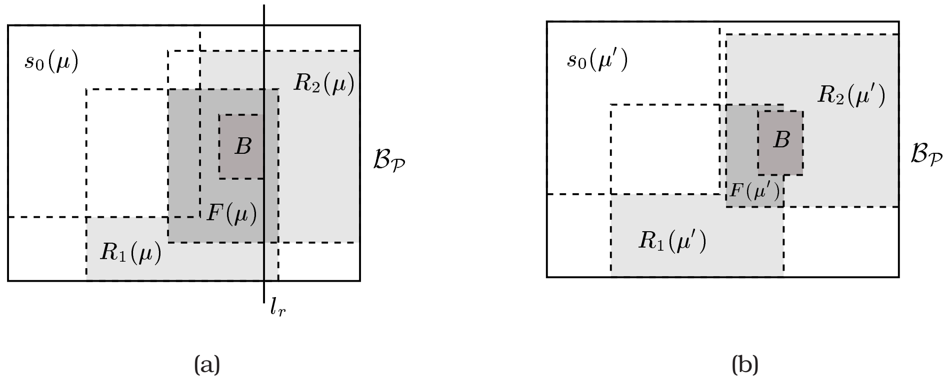


Fig. 5. Computing the magic radius μ'.

Proof. Let $n := |\mathcal{P}|$. Run Algorithm 2 where at each iteration the four radii from Lemma 19 are tested. Due to Proposition 18, each iteration takes $\mathcal{O}(n)$ time. At the end of any iteration, either the optimal solution can be computed in linear time, or at least $\lfloor \frac{n-22}{6} \rfloor$ points are discarded from $\mathcal{P}$. Hence, for $n > 27$ at least one point is discarded and the algorithm terminates. The runtime $T(n)$ can be bounded by $T(n) \leq cn + T(\lceil \frac{5n+22}{6} \rceil) < 6cn + 21c \lceil \log_2 n \rceil + d$, for some positive constants $c, d \in \mathbb{R}$. □

Theorem 21. *The rectilinear* 3*-center problem for a set* $\mathcal{P} \subset \mathbb{R}^2$ *of* n *points can be solved in* $\mathcal{O}(n)$ *time.*

Proof. For each of the four possible placements for $s_0(\varrho)$, compute the optimal type 1 covering according to Lemma 20 and the optimal type 2 covering as described in Lemma 14. Due to Proposition 11, the best solution found in any of these subcases is the overall optimal solution. □

Remark 22. Observe that there is a slight overlap in the different subcases mentioned in the proof of Theorem 21. The optimal type 2 coverings are the same, if $s_0(\varrho)$ is placed at opposite corners of $\mathcal{B}(\mathcal{P})$. Hence, there are six subcases only instead of eight.

4 Experimental Results

There are a few things to note regarding the algorithm's implementation. First, there is the algorithm for linear selection. Although this can be done in deterministic linear time as noted, the constants in this linear term are rather large. From a practical point of view the standard randomized selection algorithm that needs expected linear time seems preferable. Hence, this is what has been used here.

4.1 Test Data

In order to test the performance of an algorithm, one needs to have a set of test data. An important question in this context is, for what kind of input data one wants to evaluate the algorithm and sometimes also how to generate this data efficiently.

In this case the input consists of a set of points. So one of the most simple test data conceivable is a set of points drawn uniformly at random from the unit square. But as the number of points increases, the 3-coverings of these random point sets tend to consist of three almost coinciding squares with side length close to one. So in some sense these are very special input sets where one cannot expect to find a "nice" way to divide the points into three clusters.

Therefore we have experimented with a second type of random point sets. These consist of points drawn uniformly from three congruent squares of some fixed side length. The squares in turn are placed uniformly at random inside the unit square. These sets are also particularly useful to verify the correctness of the algorithm and its implementation, since with high probability the optimal solution is known in advance.

4.2 Heuristics

Additionally, two simple heuristics have been added to speed up the algorithm in many cases. In Step (2) of Algorithm 2, the size of s_0 is adjusted such that it contains about $\frac{1}{6}$ of the points. This leads to a rather slow convergence if the square contains many points in the optimal covering.

Heuristic 3 (Prefilter). First try to cover half of the points with s_0, that is set $k := \lfloor \frac{n}{2} \rfloor$. If the corresponding radius μ is infeasible, continue with cutoff $k = \lfloor \frac{n}{2} \rfloor$. But as soon as μ is feasible for the first time, switch to $k = \lfloor \frac{n-10}{6} \rfloor$ for the remainder of the algorithm.

A single run of Algorithm 2 just gives the optimal type 1 covering for one out of four possible placements of s_0. Additionally, there are two different optimal type 2 coverings that have to be computed. As soon as the first of these partial solutions has been obtained, the resulting radius provides an upper bound for the overall optimum. Obviously, it does not make sense to compute coverings with a radius exceeding that bound.

Heuristic 4 (Check). Check at the start of each run whether one can beat the current bound in the actual setting. Proceed only if this is possible.

Note that the test mentioned in Heuristic 4 is simply an instance of the corresponding decision problem. Obviously, neither of these heuristics changes the algorithm's asymptotic behavior. While in some cases either one of them might result in a loss of performance, this slowdown is expected to be small compared to the overall runtime. Regarding the test data as described in the previous section, one expects

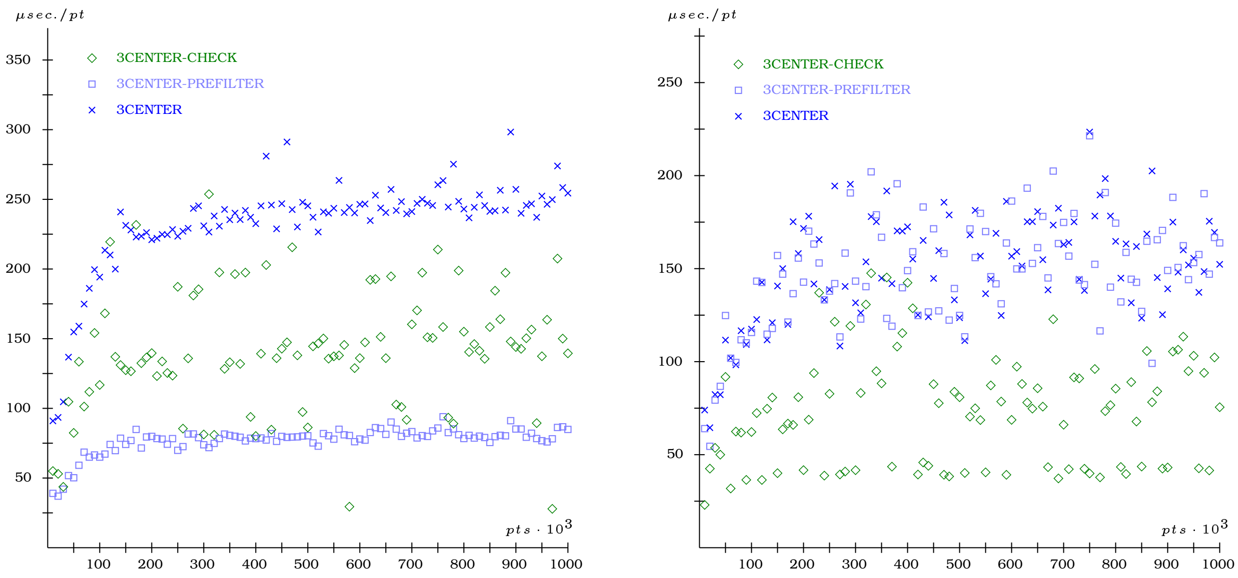
- prefiltering to improve the performance significantly for point sets drawn uniformly from the unit square,
- prefiltering to have less positive influence for point sets drawn uniformly from three clusters as the side length of the cluster squares decreases *and*
- checking to improve the performance in most cases independently from the input data.

For Heuristic 4 note that there is a gain whenever good solutions are found first. Hence, the worst case complexity can be improved by randomizing the order in which the different cases are handled. A significant loss should only occur in the rare case that the solutions are encountered in worst-to-best order.

Indeed, these expectations can be confirmed by looking at the experimental results depicted in Fig. 6. The x-axis describes the input size in units of 1000 points. The y-axis gives the corresponding runtime in microseconds per point[1] for the algorithm with and without using each of the two heuristics.

It turns out that for the clustered input data the effect of prefiltering is positive in only about half of the examples, while for the other half there is an increase in runtime. But even for very small cluster sizes (results not shown here) the slowdown is small compared to the overall runtime, such that the general use of both heuristics is justified.

[1] on an SGI Indigo2 with 175MHz MIPS R10000(IP28) processor



(a) Points from the unit square.

(b) Points from three clusters.

Fig. 6. Comparison of the heuristics for different randomly generated input sets.

4.3 Comparison to Matrix-Search based Algorithms

The CGAL [6] library contains an $\mathcal{O}(n \log n)$ implementation for computing rectilinear 2-, 3- and 4-centers based on searching in sorted matrices [12, 13]. We have compared the performance of the new 3-center algorithm to the 3-center algorithm currently in CGAL, expecting to beat it at least for sufficiently large point sets.

The results of these test runs are shown in Fig. 7. Again the x-axis describes the input size in units of 1000 points and the y-axis gives the corresponding runtime in microseconds per point. The runtime is shown for three different algorithms: the matrix search algorithm, the new 3-center algorithm, and the new 3-center algorithm with both heuristics applied.

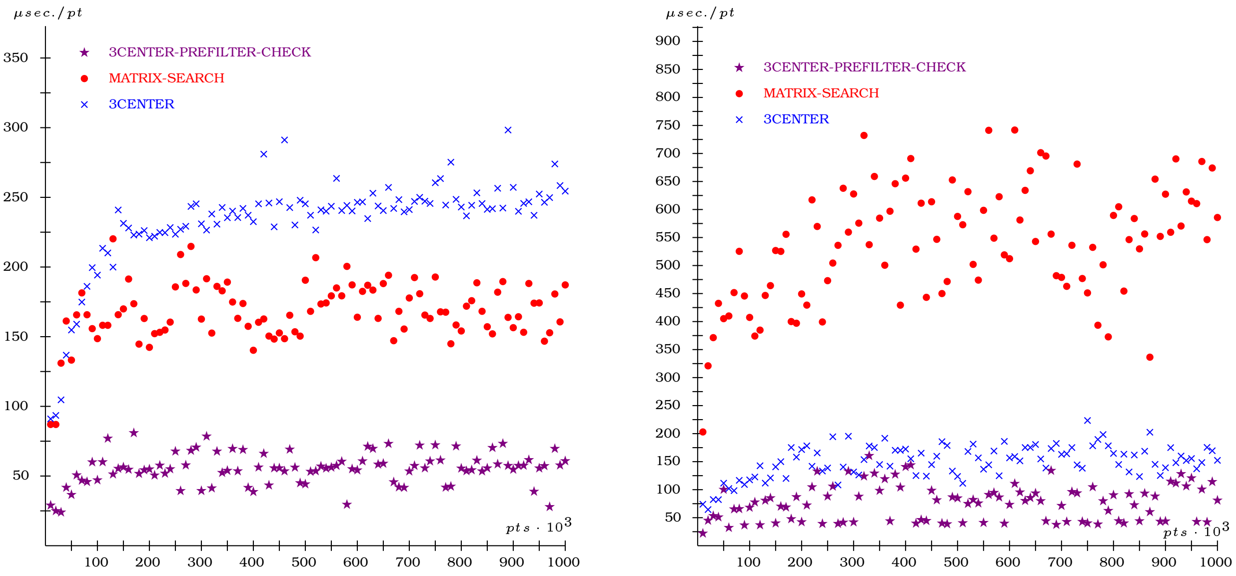
It turns out that the linear algorithm outperforms the matrix-search even for small point sets from the test data, at least as soon as both heuristics described in the previous section are applied. Without prefiltering, the matrix-search is faster on point sets drawn uniformly from the unit square. But as already noted above, these point sets are somewhat special in the sense that their 3-radius is close to the maximum possible value, half the width of their bounding box.

5 Conclusions

We have described and implemented a new linear time algorithm for the rectilinear 3-center problem and two heuristics to improve its performance in practice. A number of tests on certain randomly generated test sets have been made to compare the new algorithm to an existing $\mathcal{O}(n \log n)$ implementation based on searching sorted matrices. In all tests the linear algorithm outperformed the $\mathcal{O}(n \log n)$ algorithm with factors ranging from three to eight. The implementation appeared as part of the Computational Geometry Algorithms Library (CGAL) since Release 2.1 (January 2000).

Acknowledgments.

I thank Bernd Gärtner, Lutz Kettner, Bettina Speckmann, Emo Welzl, and the anonymous referees for their comments on preliminary versions of this paper.



(a) Points from the unit square.

(b) Points from three clusters.

Fig. 7. Comparison to a matrix-search based algorithm for different randomly generated input sets.

References

1. Agarwal, P. K., and Procopiuc, C. M. Exact and approximation algorithms for clustering. *Algorithmica 33* (2002), 201–226.
2. Agarwal, P. K., Sharir, M., and Welzl, E. The discrete 2-center problem. *Discrete Comput. Geom. 20* (1998), 287–305.
3. Assa, E., and Katz, M. J. 3-piercing of d-dimensional boxes and homothetic triangles. *Internat. J. Comput. Geom. Appl. 8* (1999), 249–260.
4. Bespamyathnikh, S. N., and Segal, M. Rectilinear static and dynamic discrete 2-center problems. In *Proc. 6th Workshop Algorithms Data Struct.* (1999), vol. 1663 of *Lecture Notes Comput. Sci.*, Springer-Verlag, pp. 276–287.
5. Blum, M., Floyd, R. W., Pratt, V., Rivest, R. L., and Tarjan, R. E. Time bounds for selection. *J. Comput. Syst. Sci. 7* (1973), 448–461.
6. CGAL. The Computational Geometry Algorithms Library. `http://www.cgal.org/`.
7. Chan, T. M. More planar two-center algorithms. *Comput. Geom. Theory Appl. 13* (1999), 189–198.
8. Drezner, Z. On the rectangular p-center problem. *Naval Res. Logist. Q. 34* (1987), 229–234.
9. Dyer, M. E. On a multidimensional search technique and its application to the Euclidean one-centre problem. *SIAM J. Comput. 15* (1986), 725–738.
10. Eppstein, D. Faster construction of planar two-centers. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms* (1997).
11. Feder, T., and Greene, D. H. Optimal algorithms for approximate clustering. In *Proc. 20th Annu. ACM Sympos. Theory Comput.* (1988), pp. 434–444.
12. Frederickson, G. N., and Johnson, D. B. Finding kth paths and p-centers by generating and searching good data structures. *J. Algorithms 4* (1983), 61–80.
13. Frederickson, G. N., and Johnson, D. B. Generalized selection and ranking: sorted matrices. *SIAM J. Comput. 13* (1984), 14–30.
14. Halperin, D., Sharir, M., and Goldberg, K. The 2-center problem with obstacles. *J. Algorithms 42*, 1 (2002), 109–134.
15. Hoffmann, M. Covering polygons with few rectangles. In *Abstracts 17th European Workshop Comput. Geom.* (2001), Freie Universität Berlin, pp. 39–42.
16. Jaromczyk, J. W., and Kowaluk, M. An efficient algorithm for the Euclidean two-center problem. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.* (1994), pp. 303–311.

17. KATZ, M. J., KEDEM, K., AND SEGAL, M. Discrete rectilinear 2-center problems. *Comput. Geom. Theory Appl. 15* (2000), 203–214.
18. KHULLER, S., AND SUSSMANN, Y. J. The capacitated k-center problem. *SIAM J. Discrete Math. 13*, 3 (2000), 403–418.
19. KO, M. T., AND CHING, Y. T. Linear time algorithms for the weighted tailored 2-partition problem and the weighted rectilinear 2-center problem under L_∞-distance. *Discrete Appl. Math. 40* (1992), 397–410.
20. KO, M. T., AND LEE, R. C. T. On weighted rectilinear 2-center and 3-center problems. *Inform. Sci. 54* (1991), 169–190.
21. KO, M. T., LEE, R. C. T., AND CHANG, J. S. An optimal approximation algorithm for the rectilinear m-center problem. *Algorithmica 5* (1990), 341–352.
22. MATOUŠEK, J., SHARIR, M., AND WELZL, E. A subexponential bound for linear programming. *Algorithmica 16* (1996), 498–516.
23. MEGIDDO, N. The weighted Euclidean 1-center problem. *Math. Oper. Res. 8*, 4 (1983), 498–504.
24. MEGIDDO, N. Linear programming in linear time when the dimension is fixed. *J. ACM 31* (1984), 114–127.
25. MEGIDDO, N., AND SUPOWIT, K. J. On the complexity of some common geometric location problems. *SIAM J. Comput. 13*, 1 (1984), 182–196.
26. MEGIDDO, N., AND ZEMEL, E. A randomized $O(n \log n)$ algorithm for the weighted Euclidean 1-center problem. *J. Algorithms 7* (1986), 358–368.
27. NUSSBAUM, D. Rectilinear p-piercing problems. In *Proceedings of the Annual International Symposium on Symbolic and Algebraic Computation* (1997), pp. 316–323.
28. SEGAL, M. On piercing of axis-parallel rectangles and rings. In *Proc. 5th Annu. European Sympos. Algorithms* (1997), vol. 1284 of *Lecture Notes Comput. Sci.*, Springer-Verlag, pp. 430–442.
29. SHARIR, M. A near-linear algorithm for the planar 2-center problem. *Discrete Comput. Geom. 18* (1997), 125–134.
30. SHARIR, M., AND WELZL, E. Rectilinear and polygonal p-piercing and p-center problems. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.* (1996), pp. 122–132.