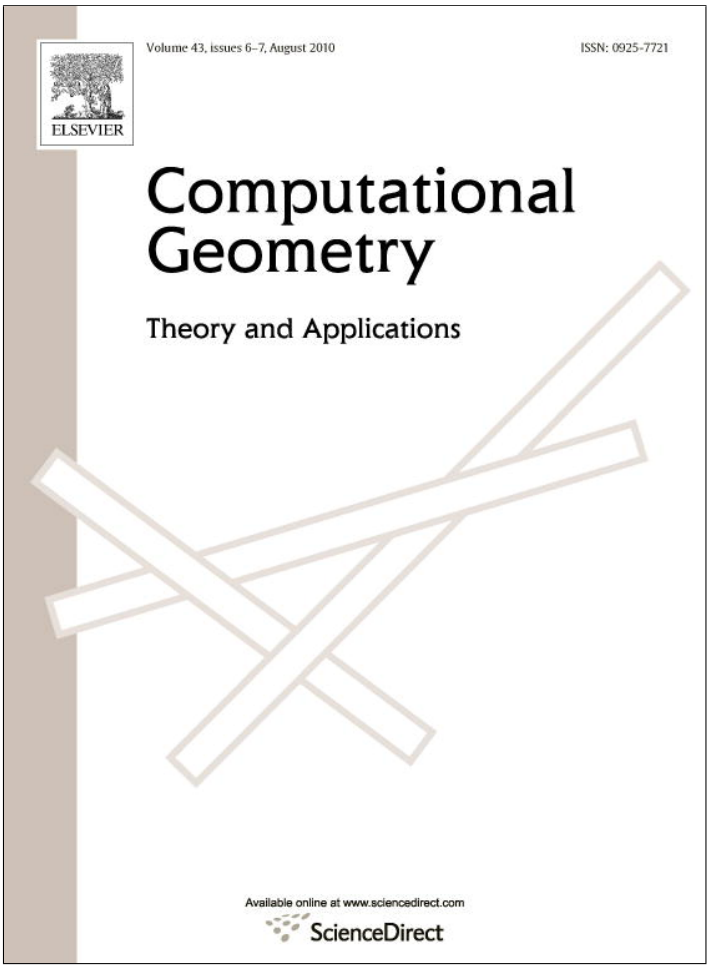

Computational Geometry 43 (2010) 572–586

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

Approximation algorithm for the kinetic robust K-center problem

Sorelle A. Friedler *,[1], David M. Mount [2]

Department of Computer Science, University of Maryland, College Park, MD 20742, USA

ARTICLE INFO

Article history:
Received 19 January 2009
Received in revised form 12 January 2010
Accepted 13 January 2010
Available online 18 January 2010
Communicated by P. Agarwal

Keywords:
Kinetic data structures
Robust statistics
Clustering
Approximation algorithms

ABSTRACT

Two complications frequently arise in real-world applications, motion and the contamination of data by outliers. We consider a fundamental clustering problem, the k-center problem, within the context of these two issues. We are given a finite point set S of size n and an integer k. In the standard k-center problem, the objective is to compute a set of k center points to minimize the maximum distance from any point of S to its closest center, or equivalently, the smallest radius such that S can be covered by k disks of this radius. In the discrete k-center problem the disk centers are drawn from the points of S, and in the absolute k-center problem the disk centers are unrestricted.

We generalize this problem in two ways. First, we assume that points are in continuous motion, and the objective is to maintain a solution over time. Second, we assume that some given robustness parameter $0 < t \leqslant 1$ is given, and the objective is to compute the smallest radius such that there exist k disks of this radius that cover at least $\lceil tn \rceil$ points of S. We present a kinetic data structure (in the KDS framework) that maintains a $(3+\varepsilon)$-approximation for the robust discrete k-center problem and a $(4+\varepsilon)$-approximation for the robust absolute k-center problem, both under the assumption that k is a constant. We also improve on a previous 8-approximation for the non-robust discrete kinetic k-center problem, for arbitrary k, and show that our data structure achieves a $(4+\varepsilon)$-approximation. All these results hold in any metric space of constant doubling dimension, which includes Euclidean space of constant dimension.

1. Introduction

In the design of algorithms for optimization problems in real-world applications, it is often necessary to consider the problem in the presence of complicating issues. We consider two such issues here. The first is the processing of *kinetic data*, that is, objects undergoing continuous motion. Applications involving kinetic data are omnipresent, including, for example, particle-based simulations in physics and monitoring the motion of moving objects such as automobiles, cell phone users, mobile sensors, or objects carrying RFID tags. The second confounding issue arises when the data is heterogeneous and the statistical trends of the majority may be obscured by the deviant behavior of a minority of points, called outliers. The objective is to produce a solution to some given optimization problem that is *robust* to corruption due to outliers.

These two issues have been considered individually in the context of kinetic data structures and robust statistics, respectively, but no published work to date has involved the combination of the two. The combination of these two issues presents unique challenges. One reason is the different effects these two issues have on the structure of algorithmic so-

* Corresponding author.
E-mail addresses: sorelle@cs.umd.edu (S.A. Friedler), mount@cs.umd.edu (D.M. Mount).

[1] The work of Sorelle Friedler has been supported in part by the AT&T Labs Fellowship Program.
[2] The work of David Mount has been supported in part by the National Science Foundation under grant CCR-0635099 and the Office of Naval Research under grant N00014-08-1-1015.

0925-7721/$ – see front matter
doi:10.1016/j.comgeo.2010.01.001

lutions. Kinetic data structures are typically concerned with handling local properties involving the interaction of a small number of objects. On the other hand, algorithmic solutions in robust statistics involve global characteristics of the data set, such as identifying which subset of points constitutes the majority. In this paper we consider a well-known clustering problem, called the k-center problem, in the context of kinetic data and outliers. We refer to it as the kinetic robust k-center problem.

Many frameworks have been proposed for handling kinetic data [26,4,38,39]. We assume a common model for processing points in motion, called *kinetic data structures* (KDS), which was proposed by Basch, Guibas, and Hershberger [5]. In this model the motion of each point is given by a piecewise function of constant algebraic degree, called a *flight plan*. KDSs track specific properties of moving points. This is done through a set of boolean conditions, called *certificates*, and a corresponding set of update rules. Certificates guarantee geometric relations necessary to a particular problem's solution, and update rules specify how to respond whenever a certificate fails. The KDS framework has become the standard approach for computing discrete structures for kinetic data sets because it provides a general and flexible framework for the development of algorithmic solutions that are both provably correct and efficient. Examples include maintaining convex hulls [5], Voronoi diagrams [2], and minimum spanning trees on geometric graphs [6].

There are four criteria under which the computational cost of a KDS is evaluated: responsiveness, efficiency, compactness, and locality [24,25]. *Responsiveness* measures the complexity of the cost to repair the solution after a certificate fails. *Efficiency* measures the number of certificate failures as compared to the number of required changes to the solution as the points move. *Compactness* measures the size of the certificate set. *Locality* measures the number of certificates in which each point participates. Guibas provides a more detailed overview of kinetic data structures in [24,25].

The other issue of interest in this paper is robustness. The study of statistical estimators that are insensitive to outliers is the domain of *robust statistics* [36]. Robust statistics have been extensively studied in mathematics, operations research, and computer science. Robustness in the presence of outliers is important, for example, when considering heterogeneous populations that contain isolated and unusual data points. It is also useful when considering business and public-service applications. Since cost is an important factor, it is desirable to provide a service to a large segment of the population, while limiting expensive service costs involving a small fraction of outliers. Charikar et al. [8] explored the robust facility location problem, which determines the locations of stores while minimizing the distance from customers to the stores and the total cost of opening facilities. In such a model it is unprofitable to open a new facility to service a small number of isolated customers. Recently, Degener et al. [14] gave a deterministic algorithm for the kinetic version of the non-robust facility location problem and Agarwal and Phillips [1] gave a randomized algorithm for the robust 2-center problem with an $O(nk^7 \log^3 n)$ expected execution time.

We consider a clustering problem that involves a combination of these two important elements. Clustering is a frequently studied problem in operations research and computer science. Common formulations include k-center, k-means, and facility location problems [30,29,17,35]. The (standard) *k-center problem* is defined as follows: Given a set of n points, find k center points that minimize the maximum distance (called the *radius*) from any point to its closest center. In the *discrete version* the centers must be drawn from the original n points. In contrast, in the *absolute version* the centers may be arbitrary points in space [30]. Unless otherwise stated, we will assume the discrete version of the problem.

Kariv and Hakimi [30] proved that the discrete and absolute versions of the k-center problem are NP-hard in a graph-theoretic context (for arbitrary k). The problem of finding a $(2-\varepsilon)$-approximation for the discrete k-center problem is NP-complete (also in a graph-theoretic context) [28,34]. The problem in the Euclidean metric cannot be approximated to within a factor of 1.822 (assuming $\mathrm{P} \neq \mathrm{NP}$) [17]. Demaine et al. [15] give algorithms for the k-center problem on planar graphs and map graphs that achieve time bounds exponential in the radius and k.

Since the k-center problem is NP-hard for arbitrary k or exponential in k for fixed k, we consider approximation algorithms. An algorithm provides a *c-approximation* to the k-center problem if the radius chosen for the k centers is no more than c times the optimal radius. Feder and Greene [17] gave a 2-approximation for the geometric k-center problem and Hochbaum and Shmoys [29] and Gonzalez [21] gave 2-approximation algorithms for the graph-theoretic version of the k-center problem, both for arbitrary k. In light of the above lower bound of $(2-\varepsilon)$, these approximation algorithms provide the best possible approximation bounds.

The *robust k-center problem* generalizes the k-center problem to handle outliers by allowing flexibility in the number of points that satisfy the distance criteria. In our formulation we are given a set of n points, an integer k, and a threshold parameter t, where $0 < t \leqslant 1$. The objective is to compute the smallest radius r such that there exist k disks of radius r that cover at least $\lceil tn \rceil$ points. The non-robust version arises as a special case, when $t = 1$.

Since the robust k-center problem is a generalization of the non-robust version, the 1.822 approximation lower bound [17] for the Euclidean context holds for the robust k-center problem as well (assuming $\mathrm{P} \neq \mathrm{NP}$). Charikar et al. [8] showed that in the graph-theoretic context, the robust k-center problem with forbidden centers (in which some locations cannot be chosen as centers), has a lower bound of $3-\varepsilon$. They also gave a 3-approximation algorithm for the robust k-center problem. Recently, Chen gave a constant factor approximation algorithm for the robust k-median problem [9].

The (non-robust) kinetic k-center problem is a generalization of the static version, so again the 1.822 approximation lower bound [17] holds. No other lower bounds are known for the kinetic problem. Gao, Guibas, and Nguyen [18] give an 8-approximation algorithm for the kinetic discrete k-center problem. Har-Peled handles the discrete and absolute kinetic k-center problems with an $O(nk)$ time algorithm, which creates a larger static set of centers that is competitive at any time [27].

The kinetic robust k-center problem has not been studied before, but many application domains involving moving points benefit from robust clustering calculations. These include the segmentation problem in vision, which attempts to separate meaningful parts of a moving image [7,16,40]; context-aware applications, which run on mobile devices that are carried by individuals and interact with the environment and each other [41]; and traffic detection and management, which we use as our main motivating example. Traffic detection has been studied extensively with tactics that include using sensors [31,23,37], knowledge-based systems [12], and individual vehicle monitoring (e.g., car GPS navigation systems) [19,3,20]. Our model assumes individual vehicle monitoring with the assumption of a flight plan provided by the navigation system and a desired number, k, of congested areas to monitor.

Throughout this paper, we will assume that the point set S resides in a space of *constant doubling dimension*. We define the *disk* of radius r centered at point u to be the set of points of S whose distance from u is less than or equal to r. A metric space is said to have constant doubling dimension if any metric disk of radius r can be covered by at most a constant number, λ, of disks of radius $r/2$. Euclidean space of constant dimension is an example. The doubling dimension d is defined to be $\log_2 \lambda$ [32]. To generalize the concept of a metric space to a kinetic context, we assume access to functions giving the distance between two points at a given time and the earliest future time at which two points will be within some given distance.

1.1. Contributions

As mentioned above, we present the first concurrent consideration of two practical domains, robust statistics and kinetic data structures, and an approximation algorithm and corresponding efficient kinetic data structure to solve the kinetic robust k-center problem. Our algorithm approximates the static k-center, kinetic k-center, and robust k-center problems as well, since all are special cases of our problem.

The input consists of a kinetic point set S in a metric space of constant doubling dimension d, the number of centers k, a robustness threshold $0 < t \leqslant 1$, and an approximation parameter $\varepsilon > 0$. Some of our complexity bounds depend on the aspect ratio of the point set, which is defined as follows in a kinetic context. Let $d_{\min}$ and $d_{\max}$ be lower and upper bounds, respectively, on the distance between any two points over the entire motion. The *aspect ratio*, denoted by α, is defined to be $d_{\max}/d_{\min}$. We obtain a $(3+\varepsilon)$-approximation for the static and kinetic forms of the robust discrete k-center problem and a $(4+\varepsilon)$-approximation for the absolute version of the robust k-center problems. Note that the first bound improves upon the 8-approximation for the kinetic discrete k-center problem as given by Gao, Guibas, and Nguyen [18] and generalizes it to the robust setting. However, due to complications arising from the need for robustness, our result assumes that k is constant, while theirs holds for arbitrary k. We improve their result for the non-robust kinetic problem for arbitrary k by showing that our data structure achieves a $(4+\varepsilon)$-approximation, while maintaining the same quality bounds as their KDS (see Section 5). To our knowledge, our kinetic robust algorithm is the first approximation algorithm for the kinetic absolute k-center problem (even ignoring robustness). We give an example in Section 3.3.4 to show that our $(3+\varepsilon)$-approximation for the robust discrete k-center problem is tight.

The KDS used by our algorithm is efficient. In Section 4.3 we will establish bounds of $O((\log\alpha)/\varepsilon^d)$ for locality and $O(n/\varepsilon^{d+1})$ for compactness. Our responsiveness bound is $O((\log n \log\alpha)/\varepsilon^{2d})$, implying that the data structure can be updated quickly. Our efficiency bound of $O(n^2(\log\alpha)/\varepsilon)$ is reasonable since the combinatorial structure upon which our kinetic algorithm is based requires $\Omega(n^2)$ updates in the worst case (even for the non-robust case) [18], so any approach based on this structure requires $\Omega(n^2)$ updates.

2. Weak hierarchical spanner

Our approach is to extend a spanner construction for kinetic data structures developed by Gao et al. [18], which they call a *deformable spanner*. This kinetic structure is defined assuming a point set S in $\mathbb{R}^d$ for any fixed d, but the construction generalizes easily to any metric space of constant doubling dimension. The spanner is based on a hierarchical clustering involving a sparse subset of points, called centers. To avoid confusion with the use of the term *center* as a cluster center, henceforth we use the term *node* for a point in the discrete hierarchy, and the term *center* when referring to the center of a disk in the solution to the k-center problem. We will use the term *point* to refer to an element of S. Each node is associated with a point of S. Because of the close relationship between nodes and the associated points, we will sometimes blur this distinction, for example, by referring both to a node u in the spanner and point u in S, or referring to the distance between two nodes (by which we mean the distance between their associated points).

Given a point set S, a *hierarchy of discrete centers* is a sequence of subsets $S_0 \supseteq S_1 \supseteq \cdots \supseteq S_m$ such that the following properties hold for $0 \leqslant i \leqslant m$:

- $S_0 = S$ and $|S_m| = 1$.
- For $i \geqslant 1$, each node of S_{i-1} is within distance 2^i of some node in S_i, the ith level of the hierarchy.
- For any two nodes $u, v \in S_i$, with associated points $u, v \in S$, $\|uv\| \geqslant 2^i$.

By definition, for each node v in level $i-1$, there exists a node u in level i such that v is within distance 2^i of u. One such node u is selected (arbitrarily) to be v's parent (and v is a *child* of u). We use other standard tree relationships

including ancestors and descendants (both of which we consider in the improper sense, so that a node is an ancestor and descendant of itself) and siblings [11]. Some properties about the deformable spanner, which follow immediately from the above properties or are proven in [18], are given below:

- The hierarchy has a height $O(\log\alpha)$.
- Any node in S_0 is within distance 2^{i+1} of its ancestor in level S_i.

Gao et al. [18] showed that the hierarchy of discrete centers could be used to define a spanner for the point set S. Given a parameter $\gamma \geqslant 1$, called the *stretch factor*, a *γ-spanner* for a point set is a graph where the points are the vertices and the edge set has the property that the shortest path length in the graph between any two points is at most γ times the metric distance between these points. Given a user-supplied parameter $c > 4$, two nodes u and v on level i of the hierarchy are said to be *neighbors* if they lie within distance $c \cdot 2^i$ of each other. For each pair of neighboring nodes in the hierarchy, an edge is created between their associated points. Gao et al. show that the resulting graph is a spanner for S, and they establish a relationship between the value of c and the resulting stretch factor. Throughout, we will assume that $c = 8$, which implies that the resulting graph is a 5-spanner. Although we will use the term *spanner* when referring to our structure, we will not be making use of spanner properties directly in our results.

The KDS presented in [18] for the deformable spanner is shown to be efficient, local, compact, and responsive. The algorithm maintains four types of certificates: parent-child certificates, edge certificates, separation certificates, and potential neighbor certificates. We will use these same certificates in our algorithm, but with different update rules.

There is one additional difference between our structure and that of Gao et al. In order to obtain our stronger approximation bounds, we will need to make a number of copies of this structure, each with slightly different parameter settings. The number of copies, which depends on the approximation parameter ε, will be denoted by $s(\varepsilon)$, or simply s whenever ε is clear from context. Its value will be given in the next section. Recall that the ith level of the hierarchy of discrete centers is naturally associated with the distance 2^i (both as the covering radius and as the separation distance between nodes). The lowest level of the hierarchy is associated with the distance $2^0 = 1$, which we call the *base distance* of the hierarchy. Each copy in our structure will employ a different base distance. In particular, for $0 \leqslant p < s$, let $b_p = (1 + \frac{p}{s})$. Observe that $1 \leqslant b_p < 2$. In our structure, the ith level of the pth spanner copy, denoted $S_i(p)$, will be associated with the distance $b_p 2^i$. The neighbors of a node are defined to be those nodes within distance $c \cdot b_p 2^i$, rather than $c \cdot 2^i$.

To simplify our algorithm presentations, unless otherwise stated, we will assume that $p = 0$, and so $b_p = 1$. There is no loss of generality in doing so, because an equivalent way of viewing the variation in the base distance is to imagine that distances have been scaled. In particular, when dealing with the pth copy, imagine that all distances have divided by b_p, and the hierarchy is then constructed on the scaled points using the default base distance of 1.

Since the lowest level of the hierarchy, S_0 is required to satisfy the requirement that the distance between any two points is at least $2^0 b_p$, it will be useful to assume that distances have been scaled uniformly so that $d_{\min} = 2$.

3. Robust *K*-center algorithm

Gao et al. [18] gave an 8-approximation for the non-robust discrete version of the kinetic k-center problem (for arbitrary k). In Section 5 we improve this to a $(4+\varepsilon)$-approximation for arbitrary k. In this section we present a $(3+\varepsilon)$-approximation algorithm for the robust discrete version of this problem and a $(4+\varepsilon)$-approximation algorithm for the robust absolute version, both for constant k. Recall that the non-robust version is a special case of the robust version (by setting $t = 1$), so these algorithms also apply to the non-robust case.

3.1. Intuitive explanation

For the sake of intuition regarding some of the more complex technical elements of our algorithm we first present the algorithm by Charikar et al. [8] for the static robust discrete k-center problem, which is the basis for our algorithm, and we explain why it cannot be applied directly in the kinetic context. Henceforth we refer to it as the *expanded-greedy algorithm*.

The algorithm is given as input a point set S of cardinality n, a number of centers k, and a robustness threshold t. Radius values are chosen in a parametric search so that all potential optimal values are considered. For a given target radius r and for each point $v \in S$, we define the *greedy disk* G_v to be a disk of radius r centered at v, and the *expanded disk* E_v to be the disk of radius $3r$ centered at v. (We sometimes let G_v and E_v represent the geometric disk and sometimes the subset of S contained within the disk. It will be clear from context which interpretation is being used.) Initially all the points of S are labeled as uncovered. The algorithm repeatedly picks the node v such that the greedy disk G_v contains the most uncovered points, and it then marks all points within the expanded disk E_v as covered. If after k iterations it succeeds in covering at least $\lceil tn \rceil$ points, it returns successfully, and otherwise it fails. The algorithm is presented in Fig. 1.

The analysis of this algorithm's approximation bound (which will be presented later in Section 3.3) uses a charging argument, where each point covered by an optimal disk is charged either to the expanded disk that covers it or to a non-overlapping greedy disk [8]. The proof relies on two main points. First, greediness implies that any optimal disk that does not overlap any greedy disk cannot cover more points than any greedy disk. Second, if an optimal disk does overlap some

```
expanded-greedy(S, k, t, r)
  n ← |S|
  C_r ← ∅   (C_r will hold the set of k centers being created for radius r)
  V ← S   (V holds the set of candidate centers)
  for each v ∈ V
    construct G_v and E_v and compute count |G_v| of uncovered points in S
      within distance r of v
  for j = 1 to k, let v_j be the v ∈ V with largest |G_v|
    add v_j to C_r and mark all points in E_v as covered
    for all v ∈ V, update |G_v|
  if at least ⌈tn⌉ points are covered, return C_r, and otherwise return "failure"
```

Fig. 1. An overview of the expanded-greedy algorithm [8] for a single radius value r.

greedy disk G_v, then the expanded disk E_v covers all the points of this optimal disk. This implies that optimal disks cannot be repeatedly "damaged" by greedy disks.

To better motivate our kinetic algorithm, it will be helpful to first consider a very simple static algorithm, which does not achieve the desired approximation bound, but we will then show how to improve it. This initial algorithm applies the expanded-greedy algorithm to individual levels of the discrete hierarchy described in Section 2. It starts at the highest level of the spanner and works down. For each level i, it runs the expanded-greedy algorithm with radius $r = 2^i$, considering just the nodes at this level as possible centers. It returns the set of centers associated with the lowest level that succeeds in covering at least $\lceil tn \rceil$ points.

There are, however, some assumptions inherent to the expanded-greedy algorithm that do not hold for this simple static algorithm. Let us consider each of these assumptions and our approach for dealing with them.

3.1.1. All important radii will be considered

The proof of the expanded-greedy algorithm relies on the possibility for the algorithm to pick a node and cover all points within the optimal radius of that node. However, in this initial algorithm, if the optimal radius is slightly larger than 2^i then our algorithm would be forced to choose the centers at the next higher level, nearly doubling the radius value.

As mentioned at the end of Section 2, we solve this problem by creating multiple spanners with base distances that vary, thus partitioning the interval between 2^i and 2^{i+1} into $O(1/\varepsilon)$ subintervals (see Section 4). The algorithm is then applied to all spanners, and the best result over all is chosen.

3.1.2. All points in S are candidate centers

Our simple static algorithm considers only nodes in level i as possible centers, and so points of S that do not reside on level i are excluded from consideration. If some of these excluded centers are in the optimal solution, then the algorithm might need to substitute a node at level i at distance up to 2^{i+1} from an optimal center, which would require the need for a larger radius.

It would be unacceptably slow in the kinetic context to consider all the points of S. Our solution instead is to take candidate centers from level $i - l - 1$, for a suitably chosen l (whose value will depend on ε). We will show that, for each optimal center, there is at least one candidate point that is close enough to enable us to obtain our approximation bounds. We are now able to cover all of the points covered by an optimal solution since the optimal center is a descendant of some center in this lower level.

3.1.3. $|G_v|$ and $|E_v|$ are known exactly

For the static case, the algorithm of Charikar et al. [8] accurately counts the number of points within the greedy and expanded radii of each node. However, maintaining these counts in a kinetic context would require keeping certificates between each point in S and all potential covering centers. This would increase the compactness and locality complexities (presented later in Section 4.3) by an unacceptable amount.

The hierarchical spanner structure allows us to count the number of points in a *fuzzy greedy disk* in which all points within some inner distance are guaranteed to be counted and no points outside of some outer distance are counted. We call this *range sketching*. Due to fuzziness, some of the counted points may lie outside the greedy radius, but we can increase the expanded radius slightly so that any optimal disks affected by this fuzziness are still fully covered by the expanded disk.

3.2. Preconditions

In this section we describe the data that we maintain in our kinetic algorithm in order to produce an approximate solution to the robust k-center problem. It will simplify the presentation to assume for now that the points are static and rely on the description of the kinetic data structure in Section 4 for proof that these values are maintained correctly in the kinetic context.

Recall that our construction involves multiple copies of the spanner using various base distances. From the real parameter $\varepsilon > 0$ we derive two additional integer parameters $s(\varepsilon) = \lceil 10/\varepsilon \rceil$ and $l(\varepsilon) = 4 - \lfloor \log_2 \varepsilon \rfloor$ (abbreviated respectively as s and l whenever ε is clear). These values will be justified in the analysis appearing in the proof of Theorem 3.2. The value

s represents the number of spanners we maintain, and l determines the number of levels of the hierarchy that we will descend at each step of the algorithm in order to find candidate centers. Observe that $s = O(1/\varepsilon)$ and $l = \log_2(1/\varepsilon) + O(1)$.

Our algorithm depends on a number of radius values, each of which is a function of the level i, the spanner copy $0 \leqslant p < s(\varepsilon)$, and $l(\varepsilon)$. Given ε, i, and p, we define the *greedy radius* and the *expanded radius* to be, respectively

$$g_i(\varepsilon, p) = 2^i\left(1 + \frac{p}{s}\right)\left(1 + 3 \cdot 2^{-l}\right) \quad \text{and} \quad e_i(\varepsilon, p) = 3g_i(\varepsilon, p).$$

We also define two slightly smaller radii

$$g_i^-(\varepsilon, p) = 2^i\left(1 + \frac{p}{s}\right)\left(1 + 2^{-l}\right) \quad \text{and} \quad e_i^-(\varepsilon, p) = 3g_i^-(\varepsilon, p).$$

When ε and p are clear from context, we abbreviate the values as g_i, e_i, g_i^-, and e_i^-, respectively. Given the important role of level $i - l - 1$ in our constructions, when l is clear from context, we define $i^- = \max(0, i - l - 1)$, and then use i^- in these contexts.

Our algorithm maintains the following information:

- For each node u at level i of the spanner, we maintain:
  - The point of S associated with u, and conversely the nodes associated with each point of S.
  - The parent, children, and neighbors of u.
  - The number of points of S that are descendants of u.
- For each node u at level i^- of the spanner, we maintain:
  - The number of points lying approximately within distance g_i of u.
  - The number of points lying approximately within distance e_i of u.

  (The sense of approximation will be defined formally in Section 3.3.)
- For each level i in the discrete hierarchy we maintain:
  - A priority queue associated with level i storing the nodes of S_{i^-} ordered by the counts of points within distance g_i (approximately) as described above. (The use of this priority queue will be clarified in Section 4.)

3.3. The discrete problem

Recall that our algorithm takes as input the set S of n points and additional parameters ε (approximation parameter), k (number of centers), and t (robustness threshold). Also recall that α denotes S's aspect ratio bound. The algorithm makes use of two parameters s and l, which are both functions of ε. It returns a set of k centers chosen from S for a $(3 + \varepsilon)$-approximation of the optimal solution to the kinetic, robust k-center problem.

3.3.1. Algorithm overview

The algorithm is applied to all s spanners in our structure. For each spanner, it applies a binary search over its levels, to determine the smallest covering radius. At each level i, the k best centers for that level are calculated using the per-level subroutine described later in this section. If this algorithm returns in failure (meaning that it failed to cover at least $\lceil tn \rceil$ points of S), the binary search continues by considering higher levels of the spanner (larger covering radii); otherwise, it continues to search through the lower levels (smaller radii). On termination of the binary search, the k centers resulting from the search are stored as representatives for that spanner. The k centers with minimum radius e_i out of all s spanners is output as the final solution.

3.3.2. Range sketching

In order to maintain the counts described as necessary preconditions, we need to efficiently count the number of points of S within a fuzzy disk, which we call a *range-sketch query*. We are given a pair of concentric disks (B^-, B), where $B^- \subseteq B$, and returns a count including all points within B^- and no points outside of B [13]. Given a node v at level i^-, let G_v and G_v^- denote the disks centered at v of radii g_i and g_i^-, respectively, and let E_v and E_v^- denote the disks centered at v of radii e_i and e_i^-, respectively. In our algorithm we will apply range-sketch queries of two types, (G_v^-, G_v) and (E_v^-, E_v). The answer to the query (B^-, B) will be represented as a collection of spanner nodes, all from the same level of the spanner, where the desired count is the total number of points descended from these nodes. This collection of nodes will be denoted by $\mu(B)$. See Fig. 2 for an illustration.

We answer a range-sketch query by first identifying an easily computable superset of nodes covering the query region, and then pruning this to form the desired set of nodes. To determine this superset of $\mu(G_v)$ (or $\mu(E_v)$) we first develop the following lemmas. Recall from Section 2 the concept of a node's neighbors, and let $c = 8$ denote the parameter used in the definition. More formally, given a subset U of nodes at some level of the spanner, let

$$N(U) = \bigcup_{u \in U} \big(\{u\} \cup \mathit{neighbors}(u)\big).$$

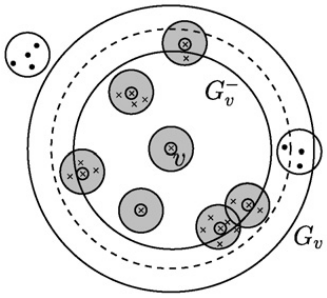

Fig. 2. For $v \in S_{i^-}$ the range-sketch query returns $\mu(G_v)$, all the nodes of S_{i^-} that lie within G_v^- and no nodes with any descendants that lie outside of G_v. These nodes are circled and disks are shaded. The points drawn as "×" are in $D(\mu(G_v))$ and are counted for the priority queue. The dashed circle has radius $(g_i^- + g_i)/2$.

Given a node v and $h \geqslant 0$, we define its *h-fold neighbor set* to be:

$$N^{(h)}(v) = \begin{cases} \{v\} & \text{if } h = 0, \\ N(N^{(h-1)}(v)) & \text{otherwise.} \end{cases}$$

Given a set of nodes U, let $D(U)$ denote the set of descendants of those nodes, and let $D^{(h)}(v) = \bigcup_{u \in N^{(h)}(v)} D(u)$.

Lemma 3.1. *For any node v in S_i and any $h \geqslant 1$, all the points of S that lie within distance $h \cdot 2^{i+3} - 2^{i+1}$ of v are in $D^{(h)}(v)$.*

Proof. Under our assumption that $c = 8$, $N(v)$ contains all the nodes in level i that are within distance $c \cdot 2^i = 2^{i+3}$ of v. (Recall that we consider the case $p = 0$.) By induction, $N^{(h)}(v)$ contains all the nodes in level i that lie within distance $h \cdot 2^{i+3}$ of v, and thus, any node u in level i that is not in this set is at distance greater than $h \cdot 2^{i+3}$ from v. Recall that, from basic spanner properties, all of u's descendants are within distance 2^{i+1} of u. It follows that $D^{(h)}(v)$ contains all the points of S within distance $h \cdot 2^{i+3} - 2^{i+1}$ of v. □

The above lemma provides a way to determine a set of nodes that cover all the points of S lying within a given distance of a given node. Using this, we can identify a set of nodes at level i^- whose descendants contain a superset of the points for the range-sketch queries of interest to us.

Lemma 3.2. *Let $h(m) = m(2^{l-2} + 1)$. Then for any node v in S_{i^-}:*

(i) *The descendants of $N^{h(1)}(v)$ contain all points within distance g_i of v.*
(ii) *The descendants of $N^{h(3)}(v)$ contain all points within distance e_i of v.*
(iii) *The descendants of $N^{h(4)}(v)$ contain all points within distance $e_i + g_i$ of v.*

Proof. By straightforward manipulations (and under our assumption that $p = 0$), each distance can be rewritten as follows:

(i) $g_i = 2^i(1 + 3 \cdot 2^{-l}) = (2^{l-2} + 1)2^{(i^- + 3)} - 2^{(i^- + 1)}$.
(ii) $e_i = 3 \cdot 2^i(1 + 3 \cdot 2^{-l}) < 3(2^{l-2} + 1)2^{(i^- + 3)} - 2^{(i^- + 1)}$.
(iii) $e_i + g_i = 4 \cdot 2^i(1 + 3 \cdot 2^{-l}) < 4(2^{l-2} + 1)2^{(i^- + 3)} - 2^{(i^- + 1)}$.

The proof follows from Lemma 3.1 applied at level i^-. □

We will apply Lemma 3.2 as a subroutine in our range-sketching procedure. Using the lemma, we first identify a set of nodes whose descendants provide a superset of the points that might contribute to the query result (depending on the radius of interest, g_i, e_i, or $g_i + e_i$), and we then prune this set by eliminating those nodes whose covering disk lies entirely outside the inner disk. To see how to perform this pruning, recall that if u is a node at level i^-, its descendants all lie within distance $2^{(i^- + 1)} = 2^{i-l}$ of u. (Note that if $i^- = 0$, then the only descendant is u itself.) Thus, if $\|uv\| > (g_i^- + g_i)/2$, then (under our assumption that $p = 0$) it is easy to verify (by the definitions of g_i and g_i^-) that every descendant u' of u satisfies

$$\|u'v\| > \frac{g_i^- + g_i}{2} - 2^{i-l} = g_i^- + \frac{g_i - g_i^-}{2} - 2^{i-l} = g_i^-,$$

and so u' may be omitted from the range-sketch result. Conversely, if $\|uv\| \leqslant (g_i^- + g_i)/2$, then it is easy to verify that every descendant u' of u satisfies $\|u'v\| \leqslant g_i$, and so u' may contribute to the range-sketch result. Given this observation, the code is given in Fig. 3 for the special case of the greedy disk G_v. It returns a set $\mu(G_v)$ of nodes whose descendants

```
range-sketch(node v, level i)
  sum ← 0
  U ← result of Lemma 3.2 part (i) applied to v
  for each u ∈ U
    if ‖uv‖ > (g_i^- + g_i)/2 then U ← U \ u
    else sum ← sum + |D(u)|
  return (U, sum)
```

Fig. 3. The range-sketch and counting subroutine for the $\mu(G_v)$ query. To answer the $\mu(E_v)$ query, all references to $\mu(G_v)$ change to $\mu(E_v)$, g_i^- and g_i to e_i^- and e_i respectively, and Lemma 3.2 part (ii) is used. We will call the range-sketching routine shown here *range-sketch*$_G$ and the one created through these substitutions *range-sketch*$_E$.

```
update-greedy-disks(node v, level i)
  U ← result of Lemma 3.2 part (iii) applied to v
  (μ(E_v), |E_v|) ← range-sketch_E(v, i)
  for each u ∈ U
    if ‖uv‖ ⩽ e_i + g_i
      (μ(G_u), |G_u|) ← range-sketch_G(u, i)
      for each w ∈ μ(E_v)
        if w is unmarked and w ∈ μ(G_u)
          |G_u| ← |G_u| − |D(w)|
          update u's position in the priority queue
```

Fig. 4. Subroutine to update greedy disk counts that calls the range-sketch subroutine shown in Fig. 3.

```
per-level-subroutine(S, k, t, ε, p, level i)
  n ← |S|
  C_{p,i} ← ∅   (C_{p,i} = {k centers being created for spanner p and level i})
  V ← S_{i^-}(p)   (V = {candidate centers}, S_{i^-}(p) = {level i^- of spanner p})
  for each v ∈ V
    (μ(G_v), |G_v|) ← range-sketch_G(v, i)
    (μ(E_v), |E_v|) ← range-sketch_E(v, i)
  for j = 1 to k
    let v_j be the v ∈ V with the largest |G_v|
    C_{p,i} ← C_{p,i} ∪ {v_j}
    update-greedy-disks(v_j, i)
    for each uncovered w ∈ μ(E_v), mark w as "covered"
  if at least ⌈tn⌉ points are covered return (C_{p,i}, e_i) otherwise return "failure."
```

Fig. 5. The per-level subroutine for spanner p and level i of the kinetic robust k-center algorithm. This subroutine calls the range-sketch and update-greedy-disk subroutines shown in Figs. 3 and 4 respectively.

satisfy the requirements of the range sketch. The desired count is the sum of the numbers of descendants of these nodes, $|D(\mu(G_v))|$. The procedure returns both the set of nodes and the sum.

Recall from our earlier description of the expanded-greedy algorithm, that whenever a center v is added to the solution at level i, the points lying within the expanded disk E_v are marked as covered. In a kinetic setting it is too expensive to mark these points explicitly. Our approach instead will be to modify the counts associated with each greedy disk that overlaps E_v. In particular, we apply range sketching to determine the nodes u in level i^- whose greedy disk G_u overlaps E_v, and for each unmarked node w in their common intersection, we decrease $|G_u|$ by the weight $D(w)$. The procedure is given in Fig. 4. To prevent nodes from being counted twice, we mark all these nodes w after E_v has been processed (see Fig. 5).

3.3.3. Main subroutine and analysis

We now have the tools needed to introduce the main subroutine for the algorithm. Recall that the input consists of the point set S and parameters k, t, ε. The quantities s and l depend on ε and represent the number of spanners and the number of levels of resolution, respectively. The current spanner is indexed by $0 \leqslant p < s$, and the current level of the discrete hierarchy is i. The per-level subroutine (presented in Fig. 5) calculates the candidate list of k centers for a given spanner p and level i. It is called from the algorithm overview presented earlier.

Before giving the kinetic version of the algorithm, we first describe the static version, and we focus on just one stage of the algorithm. In the static context this algorithm requires preprocessing to create the priority queue and perform range sketching to determine initial counts for G_v and E_v in all levels of all spanners. We comment that this can be done in time $O((1/\varepsilon)^d n \log n \log \alpha)$ since there are n points, priority queue insertions take time $O(\log n)$, centers are calculated for $O(\log \alpha)$ levels, and range sketching takes time $O(1/\varepsilon^d)$ as shown by the following lemma.

Lemma 3.3. *Range-sketch queries for level i involving any of the distances g_i, e_i or $g_i + e_i$ can be answered in time $O(1/\varepsilon^d)$.*

Proof. The running time of a range-sketch query for a node v is dominated by the time needed to compute the set $U = N^{h(m)}(v)$ of nodes at level i^- identified by Lemma 3.2. By Lemma 3.1, the distance from v to any of these nodes is at most

$$h(4)2^{(i^-+3)} - 2^{(i^-+1)} \leqslant 4\big(2^{l-2}+1\big)2^{(i^-+3)} \leqslant 16 \cdot 2^{(i^-+l)}.$$

Recall that $i^- = \max(0, i-l-1)$. If $i^- = 0$, the distance to any of the identified nodes is $O(2^l)$. If $i^- = i-l-1$, the distance is $O(2^i)$.

By definition of the hierarchy, the nodes of level i^- are separated from each other by a distance of at least 2^{i^-}. By a standard packing argument, this implies that the number of such nodes within any disk of radius r is at most $O((1+r/2^{i^-})^d)$, where d is the dimension. If $i^- = 0$, it follows that the number of nodes identified in Lemma 3.2 is $O((1+2^l/2^0)^d)$. On the other hand, if $i^- = i-l-1$, the number of nodes is $O((1+2^i/2^{i-l-1})^d)$. In either case, the number of nodes is clearly $O(2^{l\cdot d})$. Since $l = \log_2(1/\varepsilon) + O(1)$, it follows that the number of nodes is $O(1/\varepsilon^d)$.

The nodes of U are determined by computing the $h(m)$-fold neighbor set of v. We can do this by applying $h(m)$ levels of a breadth-first search to the graph of neighbors. The time to do this is proportional to the product of the number of nodes visited and their degrees. Gao et al. [18] show that each node has degree at most $(1+2c)^d - 1$. By our assumptions that $c = 8$ and d is fixed, this is $O(1)$. Thus, the total range-sketch query time is proportional to $|U|$, which is $O(1/\varepsilon^d)$. □

Theorem 3.1. *After preprocessing has completed, our algorithm takes time $O(k(\log n \log\log\alpha)/\varepsilon^{2d})$ per spanner, which is $O((\log n \log\log\alpha)/\varepsilon^{2d})$ under our assumption that k is a constant.*

Proof. We first show that, for a single level of a single spanner, the per-level subroutine takes time $O(k(\log n)/\varepsilon^{2d})$. To see this, observe that it performs k iterations. During each iteration, it updates the counts for $O(1/\varepsilon^d)$ greedy disks (those that overlap with disk E_j) based on the $O(1/\varepsilon^d)$ nodes in level i^- that are contained in each greedy disk. Each update also involves adjusting the position of an entry in a priority queue holding at most n points, which can be done in $O(\log n)$ time.

This per-level subroutine is invoked $O(\log\log\alpha)$ times per spanner, since there are $O(\log\alpha)$ levels in the discrete hierarchy, over which the binary search is performed to determine the best radius. Thus, the total time required to update any one spanner is $O(k(\log n \log\log\alpha)/\varepsilon^{2d})$. Since k is a constant, our algorithm takes time $O((\log n \log\log\alpha)/\varepsilon^{2d})$. □

We will now establish the approximation bound of $3+\varepsilon$.

Theorem 3.2. *Let r_{opt} be the optimal radius for the discrete robust k-center solution for S, and let r_{apx} be the radius found by our algorithm. Then for any $0 < \varepsilon \leqslant 1$, we have $r_{\text{apx}} \leqslant (3+\varepsilon)r_{\text{opt}}$.*

Proof. Let $v_1, \ldots, v_k$ denote the k optimal centers. We may express the optimal radius value, r_{opt}, as $2^i + x$ for some integer i and $0 \leqslant x < 2^i$. Let $p = \lceil sx/2^i \rceil$. If $p = s$, set $i \leftarrow i+1$ and $p = 0$ (effectively rounding up to the next spanner copy). Clearly, $0 \leqslant p < s$, and $\frac{p-1}{s}2^i < x \leqslant \frac{p}{s}2^i$.

We first show that it is *possible*, given the information we maintain and the algorithm we use, for us to cover as many points as are covered by the optimal solution. For $1 \leqslant j \leqslant k$, let O_j denote the *optimal disk* of radius r_{opt} centered at v_j. Let u_j denote the ancestor of v_j in level i^-. (If $i^- = 0$, then $u_j = v_j$.) By the basic properties of the discrete hierarchy we have

$$\|u_j v_j\| \leqslant 2^{(i^-+1)} \leqslant 2^{(i-l)}.$$

The node u_j will be considered by our algorithm (during the processing of spanner copy p and level i). Let G_j^- denote the disk of radius $g_i^-(\varepsilon, p)$ centered at u_j. We assert that every point lying within O_j will be included in the range-sketch count for u_j. To see this, let w be a point of S lying within O_j, that is, $\|v_j w\| \leqslant r_{\text{opt}}$. By the triangle inequality we have

$$\begin{aligned}\|u_j w\| &\leqslant \|u_j v_j\| + \|v_j w\| \leqslant 2^{(i-l)} + r_{\text{opt}} < 2^{(i-l)} + \left(2^i + 2^i\frac{p}{s}\right)\\ &= 2^i\left(2^{-l} + \left(1+\frac{p}{s}\right)\right) \leqslant 2^i\left(1+\frac{p}{s}\right)\left(1+2^{-l}\right)\\ &= g_i^-(\varepsilon, p).\end{aligned}$$

Therefore w lies within G_j^-, the inner radius for the range-sketch query, and so it must be included among the points counted in the range-sketch query for u_j. In summary, for each optimal disk O_j, there is a point u_j at level i^- of spanner copy p whose range sketch covers a superset of $S \cap O_j$.

To establish the approximation bound, let $u_1, \ldots, u_k$ denote the nodes chosen by our algorithm when run at level i of spanner copy p. (These are generally different from the u_j's described in the previous paragraph.) Let G_j and E_j denote the disks centered at u_j of radii $g_i(\varepsilon, p)$ and $e_i(\varepsilon, p)$, respectively. We will show that the expanded disks centered at these points *will* cover at least as many points as the optimal solution, which is as least $\lceil tn \rceil$. Since the algorithm returns the smallest expanded radius that (approximately) covers at least $\lceil tn \rceil$ points, this implies that $r_{\text{apx}} \leqslant e_i(\varepsilon, p)$.

Given a disk D, let $|D|$ denote the number of points of S contained within it. We will show that, for $0 \leqslant j \leqslant k$, $|O_1 \cup \cdots \cup O_j| \leqslant |E_1 \cup \cdots \cup E_j|$. Our proof is similar to that of Charikar et al. [8], and is based on an argument that charges each

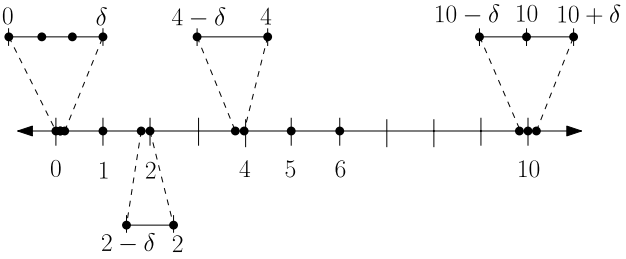

Fig. 6. An example for which our algorithm gives an approximation ratio of $3-\varepsilon$, where $k=2$ and $\lceil tn\rceil = 11$.

point covered in the optimal solution to a point covered by the solution produced by our algorithm. The proof proceeds by induction on j. The basis case is trivial, since for $j=0$, both sets are empty. Assuming by induction that $|O_1\cup\cdots\cup O_{j-1}|\leqslant |E_1\cup\cdots\cup E_{j-1}|$, we consider what happens after the next center u_j is added to the approximate solution. We consider two cases:

- If G_j intersects any of the $k-(j-1)$ remaining optimal disks, define O_j be any such optimal disk. Any point w of O_j is within distance $g_i(\varepsilon,p)+2r_{\text{opt}}$ of u_j. It is easy to verify that $r_{\text{opt}}\leqslant g_i(\varepsilon,p)$, and therefore $\|u_jw\|\leqslant 3g_i(\varepsilon,p)=e_i(\varepsilon,p)$. Thus, E_j covers all the points of O_j. We charge each point in O_j to itself.
- If G_j does not intersect any of the remaining O_j, let O_j be the remaining optimal disk covering the greatest number of points of S. By our earlier remarks, there exists a node in level i^- whose range-sketch count includes all the points of O_j. Since G_j was chosen greedily to maximize the number of unmarked points it covers, it follows that the number of unmarked points covered by G_j is at least $|O_j|$. We charge each point in O_j to an unmarked point in G_j.

Each time a point is charged, it is charged either to itself or to some point in a greedy disk G_j that is disjoint from all remaining optimal disks, and therefore, each point is charged at most once. It follows that $|O_1\cup\cdots\cup O_k|\leqslant|E_1\cup\cdots\cup E_k|$.

To complete the analysis of the approximation bound, it suffices to show that $r_{\text{apx}}/r_{\text{opt}}\leqslant 3+\varepsilon$. As observed earlier, we have

$$r_{\text{apx}}\leqslant e_i(\varepsilon,p)\quad\text{and}\quad r_{\text{opt}}=2^i+x>2^i\left(1+\frac{p-1}{s}\right).$$

Thus, we have

$$\begin{aligned}\frac{r_{\text{apx}}}{r_{\text{opt}}}&<\frac{e_i(\varepsilon,p)}{2^i(1+\frac{p-1}{s})}=\frac{3\cdot 2^i(1+\frac{p}{s})(1+3\cdot 2^{-l})}{2^i(1+\frac{p-1}{s})}\\&=3\left(1+\frac{1}{s+p-1}\right)\left(1+3\cdot 2^{-l}\right)\leqslant 3\left(1+\frac{1}{s-1}\right)\left(1+3\cdot 2^{-l}\right).\end{aligned}$$

Under our assumptions that $s=\lceil 10/\varepsilon\rceil$, $l=4-\lfloor\log_2\varepsilon\rfloor$, and $\varepsilon\leqslant 1$, it follows easily that $s-1\geqslant 9/\varepsilon$, and $3\cdot 2^{-l}\leqslant 3\varepsilon/16\leqslant\varepsilon/5$. Therefore, we have

$$\frac{r_{\text{apx}}}{r_{\text{opt}}}<3\left(1+\frac{\varepsilon}{9}\right)\left(1+\frac{\varepsilon}{5}\right)\leqslant 3\left(1+\frac{\varepsilon}{3}\right)=3+\varepsilon,$$

which completes the proof. □

The assumption that $\varepsilon\leqslant 1$ in the statement of the theorem is a technicality. The analysis may be modified to work for any constant value of ε.

3.3.4. Tightness of the approximation ratio

In this section, we present an example that demonstrates that our $(3+\varepsilon)$-approximation ratio for the discrete k-center problem is nearly tight. In particular, given any sufficiently small $\varepsilon>0$, we shall show that our algorithm achieves an approximation ratio of $3-\varepsilon$ on this example. Let $\delta=\varepsilon/6$, and consider the set of 14 points illustrated in Fig. 6. This point set consists of a collection of nine clusters placed on the real line, where each cluster contains from one to four points, each lying within distance δ of some integer point. Let $k=2$ and $\lceil tn\rceil=11$. It is easy to verify that the optimal radius is $r_{\text{opt}}=1+\delta$, which is achieved by placing centers at positions 1 and 5, so that the two disks centered at these points cover the $7+4=11$ points clustered about $\{0,1,2\}$ and $\{4,5,6\}$, respectively.

We will establish our bounds under the most favorable assumptions for the approximation algorithm. In particular, we assume that the approximation algorithm is free to select *any* point as a candidate center (not just the nodes in level i^-). We assume that the base distance for the hierarchy of discrete centers has been chosen so that *any* desired radius arises as the value of the greedy radius, $g_i^-(\varepsilon,p)$, for some level i of some spanner copy p. Finally, we assume that the counts returned

by range-sketching algorithm are *exact*. Thus, relaxing any of these restrictions in our algorithm will not significantly affect the tightness of the approximation bound.

We assert that, even under these favorable assumptions, $r_{\text{apx}} \geqslant 3(1-\delta)$. To see this, suppose not. Letting p and i denote, respectively, the spanner copy and level that produce this value, we have $e_i(\varepsilon, p) = r_{\text{apx}} < 3(1-\delta)$. For all sufficiently small δ, this is less than $4-2\delta$. This implies that each expanded disk covers at most the single cluster containing its center and the clusters about the three consecutive integer points on either side of it. We also have

$$g_i^-(\varepsilon, p) < g_i(\varepsilon, p) = \frac{e_i(\varepsilon, p)}{3} < 1-\delta.$$

Since two points from different clusters are separated by a distance of at least $1-\delta$, each greedy disk covers only a single cluster. Since the cluster near 0 has the most points (four), some point of this cluster will be chosen first. The expanded disk centered here covers only the seven points in the clusters near 0, 1, and 2. The next center to be chosen is the cluster near 10 having three points. The expanded disk is not large enough to include any other points. Thus, the algorithm succeeds in covering only $7+3=10$ points, and therefore it fails.

Since $r_{\text{apx}} \geqslant 3(1-\delta)$, we see that the approximation ratio is

$$\frac{r_{\text{apx}}}{r_{\text{opt}}} \geqslant \frac{3(1-\delta)}{1+\delta} = 3 - \frac{6\delta}{1+\delta} \geqslant 3 - 6\delta \geqslant 3 - \varepsilon,$$

as desired.

3.4. The absolute problem

Recall that in the absolute formulation the centers may be any point in space. In this section we present a $(4+\varepsilon)$-approximation algorithm for the absolute, robust k-center problem. The algorithm is the same as for the discrete problem, except that we modify the values of the radii upon which the algorithm is based. In particular, in place of $g_i(\varepsilon, p)$ and $e_i(\varepsilon, p)$, we define:

$$\hat{g}_i(\varepsilon, p) = 2g_i \quad \text{and} \quad \hat{e}_i(\varepsilon, p) = 2\hat{g}_i(\varepsilon, p).$$

We also define two slightly smaller radii for the range-sketch queries:

$$\hat{g}_i^-(\varepsilon, p) = 2g_i^-(\varepsilon, p) \quad \text{and} \quad \hat{e}_i^-(\varepsilon, p) = 2\hat{g}_i^-(\varepsilon, p).$$

Since the values of these radii have increased, we also increase the values used in various parts of Lemma 3.2 to $h(2)$, $h(4)$, and $h(6)$, respectively. By a straightforward modification of the analysis of the approximation bound given in the proof of Theorem 3.2 for the discrete case, we have the following.

Theorem 3.3. *Let r_{opt} be the optimal radius for the absolute robust k-center solution for S, and let r_{apx} be the radius found by this absolute algorithm. Then for any $0 < \varepsilon \leqslant 1$, we have $r_{\text{apx}} \leqslant (4+\varepsilon) r_{\text{opt}}$.*

Proof. The proof of Theorem 3.2 makes use of two key facts about the greedy and expanded disks. The first is that there exists a spanner copy p and a level i such that, for any optimal disk, there exists a node u in level i^- such that the associated greedy disk G_u^- of radius g_i^- contains this optimal disk. In the absolute case, the center of an optimal disk O may be at an arbitrary point of space, but by choosing any point of S that is covered by O and centering a disk O' of radius $2r_{\text{opt}}$ at this point, we see that O is contained within O'. Therefore, in our modified algorithm, there exists a node u in level i^- such that the greedy disk $\widehat{G}_u^-$ of radius $\hat{g}_i^- = 2g_i^-$ contains O', and hence contains O as well.

The second key fact used in our analysis of the discrete algorithm is that, if any optimal disk overlaps a greedy disk G_u, then the corresponding expanded disk E_u contains the optimal disk. In the absolute case, if any optimal disk O overlaps a greedy disk $\widehat{G}_u$, then every point of O lies within distance $\hat{g}_i + 2r_{\text{opt}} \leqslant 2g_i + 2g_i = 4g_i = \hat{e}_i$ of u. Therefore, $O \subseteq \widehat{E}_u$. Given these two key facts, the remainder of the proof is the same as that of Theorem 3.2, but with an appropriate adjustment of the specific values of $s(\varepsilon)$ and $l(\varepsilon)$ and with the fact that the expanded radius has increased by a factor of $\hat{e}_i/e_i = 4/3$. □

4. Kinetic spanner maintenance and quality

4.1. Certificates

The KDS algorithm of Gao, Guibas, and Nguyen [18] for the deformable spanner maintains four types of certificates. These are applied to all levels i of each spanner. A *parent-child certificate* guarantees that a node in level i is within distance 2^{i+1} of its parent. An *edge certificate* guarantees that a pair of neighboring nodes in level i lie within distance $c \cdot 2^i$ of each other (where we choose $c = 8$). A *separation certificate* guarantees that any two distinct neighboring nodes in level i are separated by a distance of at least 2^i. A *potential neighbor certificate* guarantees that two non-neighboring level-i nodes

whose parents are neighbors are separated by a distance of more than $c \cdot 2^i$. (Potential neighbor certificates are maintained so that edge certificates can be easily maintained when points move closer to each other [18].)

Recall that in our data structure we have multiple spanners with differing base distances b_p, for $0 \leqslant p < s(\varepsilon)$. As before we present the algorithm only for the simplest case of $p = 0$ and hence $b_p = 1$, but the other cases are identical up to a scaling of distances. Also recall that, for each level i, a set of k centers is maintained as well as a priority queue. The entries in this priority queue are nodes v in S_{i^-}, and the priority values are the counts of points within G_v (computed by range-sketching).

Our update rules are identical to those given in [18], with the following additions to the update rules for parent-child, edge, and separation certificates. Each rule is stated relative to some level i.

Addition of a spanner edge. Increment the counts for G_v and E_v for all neighbors indicated by Lemma 3.2 parts (i) and (ii) that are affected by this edge creation, and update the priority queue for level i appropriately.

Deletion of a spanner edge. Decrement the counts for G_v and E_v for all neighbors indicated by Lemma 3.2 parts (i) and (ii) that are affected by this edge deletion, and update the priority queue for level i appropriately.

Addition of parent-child certificate. Increment the descendant counts for all new ancestor nodes and for all counts for G_v and E_v for neighbors indicated by Lemma 3.2 parts (i) and (ii), and update the priority queue for level i appropriately.

Failure of parent-child certificate. Decrement the counts for all previous ancestor nodes and for all counts for G_v and E_v for neighbors indicated by Lemma 3.2 parts (i) and (ii), and update the priority queue for level i appropriately.

Whenever either of the count G_v changes for some node v, it may affect the k-center solution. In particular, if v's count increases and v is not a center, the k-center solution for that spanner is recalculated. Otherwise, the counts and priority queue are updated, but the solution need not be recalculated. Similarly, if v's count decreases and v is a center, the k-center solution for that spanner is recalculated, otherwise only the counts and priority queue are updated. In the cases when recalculating the solution is necessary, our static algorithm is applied. Note that as time progresses our KDS allows outlying points to become inliers and vice versa.

4.2. Preconditions

Maintaining the following preconditions ensures that the static algorithm will be applicable at any time during the motion of the points. The preconditions needed for all points are maintained by the certificates and update conditions given in the original deformable spanner [18].

For each level i in the discrete hierarchy, the following level-specific preconditions are maintained. First, as in Section 3.2, we maintain a count for each node in the discrete hierarchy of the number of points that are descendants of that node. These counts are updated whenever parent-child certificates either are created or fail. Also, for each node u at level i^-, we maintain the counts of points lying approximately within distances g_i and e_i, which are computed through the range-sketch subroutine. For each level i, we maintain the counts of the points within the fuzzy disks G_v associated with the nodes v on level i. These counts are stored in a priority queue. Whenever such a count changes (in the procedure update-greedy-disks), the priority queue is updated accordingly. If this update occurs during the course of the algorithm because a center was chosen and nodes were marked as covered, these changes are kept track of until all centers are chosen. The changes are undone in reverse order, so that the counts accurately represent the current state as the points continue to move. The preconditions for each level i are maintained according to the update rules given in Section 4.1.

4.3. Quality

In order to assure the quality of the spanner, we must reason about compactness, locality, efficiency, and responsiveness. Recall that n is the total number of points, d the dimension, and α the user-supplied upper bound on the aspect ratio. In this section we will show that compactness, locality, and efficiency are bounded by $O(n/\varepsilon^{d+1})$, $O((\log\alpha)/\varepsilon^d)$, and $O(n^2(\log\alpha)/\varepsilon)$ respectively, which match the bounds given by Gao et al. [18] up to a factor of $s = O(1/\varepsilon)$. In addition, we will establish a bound of $O((\log n \log\alpha)/\varepsilon^{2d})$ on responsiveness.

4.3.1. Compactness and locality

Compactness and locality conditions ensure that maintaining certificates for the kinetic data structure is not too costly by bounding the number of certificates. Recall that compactness bounds the total number of certificates, and locality bounds the number of certificates in which each point can participate. Since our data structure consists of $s = O(1/\varepsilon)$ copies of the spanner of [18], it follows that the compactness is larger than theirs by a factor of $O(1/\varepsilon)$ and our locality is the same. Thus we have the following.

Theorem 4.1. *Our KDS satisfies compactness and locality with $O(n/\varepsilon^{d+1})$ total certificates and $O((\log\alpha)/\varepsilon^d)$ certificates per point.*

```
non-robust(S, k, ε, α)
  for p = 0 to s − 1
    for i = ⌈log α⌉ down-to 0
      if |S_i(p)| > k
        r_p ← 2^{i+2}(1 + p/s)
        K_p ← S_{i+1}
        while |K_p| < k add an arbitrary node of S_i(p) to K_p
        break out of inner loop
  r ← min_p r_p
  output (K_p, r), each point is serviced by its ancestor in K_p
```

Fig. 7. The non-robust discrete kinetic k-center algorithm for arbitrary k. Recall that $S_i(p)$ denotes the ith level of the pth spanner.

4.3.2. Efficiency

The efficiency condition ensures that maintaining the kinetic data structure is not too expensive by bounding the number of certificate failures that can occur. This is compared to the number of required changes to the combinatorial structure of the spanner to determine the efficiency of the KDS.

Theorem 4.2. *Our KDS satisfies efficiency (with respect to the number of spanner updates) with $O(n^2(\log\alpha)/\varepsilon)$ possible certificate maintenance events.*

Proof. The deformable spanner of [18] has $O(n^2\log\alpha)$ possible maintenance events because, under pseudo-algebraic motion, events only occur when the distance between two points is at the boundary of some certificate on a given level i, namely 2^i or $c\cdot 2^i$. Since there are $O(\log\alpha)$ possible levels and $2n^2$ of these inter-point distances, there are $O(n^2\log\alpha)$ possible maintenance events [18]. Recall that there are s spanners that differ according to their base distances, but the bound on the number of maintenance events per spanner remains the same, so the total possible number of maintenance events increases by a factor of $s = O(1/\varepsilon)$. Since the spanner has not changed except for the base distance, the number of changes required by the combinatorial structure of each spanner remains $\Omega(n^2)$ [18], so any approach based on a spanner requires $\Omega(n^2)$ changes. So the kinetic data structure is efficient. □

4.3.3. Responsiveness

The responsiveness condition ensures that maintaining the kinetic data structure is not too expensive by bounding the amount of time taken to repair each failed certificate. Our spanner satisfies responsiveness with $O((\log n\log\alpha)/\varepsilon^{2d})$ time per certificate update. This time is due to the possibility that a failure or addition of a certificate could require the algorithm to be re-run and the possibility that certificates may need to be updated on each level of a single spanner.

Theorem 4.3. *Our KDS satisfies responsiveness with $O((\log n\log\alpha)/\varepsilon^{2d})$ time per certificate update.*

Proof. As shown in [18], the certificates of each copy of the hierarchical spanner can be updated in $O(1)$ or $O((\log\alpha)/\varepsilon^d)$ time depending on the specific certificate that fails. In our case, the failure or addition of a certificate could require our static algorithm to be re-run. When the priority queue is updated during the re-running, a list of changes is maintained. After k centers for a level are chosen, the priority queue is returned to its original state (the state assuming all points are uncovered). Since the changes made are undone, this increases the running time by only a constant factor. By Theorem 3.1, the solution can be recalculated in time $O(k(\log n\log\log\alpha)/\varepsilon^{2d})$. Given our assumption that k is a constant, this is $O((\log n\log\log\alpha)/\varepsilon^{2d})$.

The failure or addition of a certificate could also require points to be updated for $O(1/\varepsilon^d)$ nodes (see Lemma 3.3) on all $O(\log\alpha)$ levels. Each such update may induce a change to a priority queue entry, adding an additional factor of $O(\log n)$. Thus, the total time to repair a failed certificate is $O((\log n\log\alpha)/\varepsilon^d)$. The overall responsiveness is the maximum of these two bounds (recalculation and certificate repair), which is bounded by $O((\log n\log\alpha)/\varepsilon^{2d})$, as desired. □

5. Non-robust kinetic K-center algorithm

In this section we mention that by using our hierarchical spanner, it is possible to improve on the non-robust discrete k-center algorithm presented by Gao et al. [18]. That algorithm achieved a factor-8 approximation, and we will improve this to a $(4+\varepsilon)$-approximation (both for arbitrary k). Recall that, rather than using a single spanner, we generate $s(\varepsilon) = O(1/\varepsilon)$ spanners with differing base distances. Our approach is to run the algorithm presented by Gao et al. on all spanner copies and return the smallest radius over all these runs. The algorithm is illustrated in Fig. 7. The value of s is restricted in the proof of Theorem 5.1. For proof of maintenance under motion, we refer the reader to the proof of a similar algorithm given by Gao et al. [18].

Theorem 5.1. *Let r_{opt} be the optimal radius for k-centers chosen from the input points and r_{apx} be the radius found by our non-robust kinetic k-center algorithm, then $r_{\text{apx}} \leqslant (4+\varepsilon)r_{\text{opt}}$.*

Proof. The algorithm chooses some S_i on spanner p with associated radius $2^{i+1}(1+\frac{p}{s})$, where S_i has the minimum radius such that $|S_i| \geqslant k$. Since $2^{i+1}(1+\frac{p-1}{s}) < 2^{i+1}(1+\frac{p}{s})$, $|S_i| > k$ on spanner $p-1$. So at least two points from S_i are assigned to the same center in the optimal solution. These points are separated by a distance of at least $2^i(1+\frac{p-1}{s})$, so the optimal radius must be at least $2^{i-1}(1+\frac{p-1}{s})$. To determine the approximation ratio we consider the ratio between $r_{\text{opt}} \geqslant 2^{i-1}(1+\frac{p-1}{s})$ and $r_{\text{apx}} \leqslant 2^{i+1}(1+\frac{p}{s})$. Choosing $s(\varepsilon) \geqslant \frac{2}{\varepsilon}+\frac{1}{2}$ results in a $(4+\varepsilon)$-approximation algorithm. □

6. Conclusion

Our work demonstrates the first kinetic robust approximation algorithm for the k-center problem with an approximation ratio of $(3+\varepsilon)$ in the discrete case and $(4+\varepsilon)$ in the absolute case for fixed k. In addition, we give a $(3+\varepsilon)$-approximation algorithm for the static robust k-center problem for fixed k and a $(4+\varepsilon)$-approximation algorithm for the kinetic non-robust k-center problem for arbitrary k. Our algorithm is one of the first to tackle the conflict between the necessarily local conditions required by the kinetic data structures framework to achieve responsive update rules and globally dependent statistical information.

The KDS algorithm allows for points to change their designation as outliers or inliers over time. When considering traffic detection, outlier flexibility ensures that cars uncovered by the chosen k centers (cars that are not in traffic) may be covered at a later time (can get stuck in traffic) and vice versa. The robust kinetic k-center algorithm presented here can also be modified to handle k-center queries that have the added anonymity requirement that a given number of points must be served by a center for it to be added to the k-center solution [33]. This anonymity constraint can be satisfied by modifying the binary search through the hierarchy to find the appropriate level as presented in Section 3.3 so that all $\log\alpha$ levels are searched instead. The set of k centers at the lowest level that satisfy both the current constraints and this added anonymity constraint are output as representative for that spanner. This modification will not increase the KDS quality bounds but will increase the static algorithm runtime to $O((\log n \log\alpha)/\varepsilon^{2d})$.

Open problems in this area include revisions to the algorithm and/or kinetic framework to allow more efficient maintenance of global statistical properties under motion so that algorithms move farther away from their static counterparts and rarely need to invoke the static version of the algorithm. These modifications might additionally allow the k-center problem to be considered for arbitrary k. Currently, the algorithm assumes a fixed k, since allowing k to be arbitrary would yield an update time of $O(k(\log n \log\alpha)/\varepsilon^{2d})$ for the kinetic data structure's responsiveness. This update time is caused by re-running the static algorithm when changes to the set of centers are suspected. Due to the global nature of the k-center problem and the sequential incremental nature of our algorithm, small changes to the set of centers cannot be fixed by local incremental manipulations of the solution. Removing the restriction on k would require a different, non-sequential local solution.

In addition, it would be interesting to remove the dependence on the aspect ratio from the algorithm's time bounds. This dependence is due to the structure of the deformable spanner. Cole, Gottlieb, and Roditty [10,22] have worked on a dynamic hierarchical spanner that does not depend on the aspect ratio. Use of this spanner does not immediately yield a kinetic algorithm with no aspect ratio dependence. The Cole and Gottlieb [10] spanner makes use of path compression so that trivial spanner paths in which a point appears in multiple levels of the hierarchy without any children are represented as only the top and bottom of the path (Cole and Gottlieb call these jumps). However, the robust kinetic k-center algorithm requires a maintained priority queue for each level containing weights for each point at the level. Maintaining these priority queues would negate the space advantages of the spanner's path compression. Thus a significantly different strategy is needed to remove the aspect ratio dependence.

Acknowledgements

The authors thank anonymous reviewers for their helpful and detailed comments.

References

[1] P.K. Agarwal, J.M. Phillips, An efficient algorithm for 2D Euclidean 2-center with outliers, in: Proc. of the 16th European Symposium on Algorithms, 2008, pp. 64–75.
[2] G. Albers, L.J. Guibas, J.S.B. Mitchell, T. Roos, Voronoi diagrams of moving points, International Journal of Computational Geometry Applications 8 (1998) 365–380.
[3] N.J. Alewine, J.C. Colson, A.P. Ittycheriah, S.H. Maes, P.A. Moskowitz, Automated traffic mapping, U.S. Patent 6150961, filed November 24, 1998, and issued November 21, 2000.
[4] M.J. Atallah, Some dynamic computational geometry problems, Computers & Mathematics with Applications 11 (12) (1985) 1171–1181.
[5] J. Basch, L.J. Guibas, Data structures for mobile data, Journal of Algorithms 31 (1) (April 1999) 1–28.
[6] J. Basch, L.J. Guibas, L. Zhang, Proximity problems on moving points, in: Proc. of the 13th ACM Symposium on Computational Geometry, 1997, pp. 344–351.
[7] M.M. Chang, A.M. Tekalp, M.I. Sezan, Simultaneous motion estimation and segmentation, IEEE Trans. on Image Processing 6 (9) (1997) 1326–1333.
[8] M. Charikar, S. Khuller, D.M. Mount, G. Narasimhan, Algorithms for facility location problems with outliers, in: Proc. of the 12th ACM Symposium on Discrete Algorithms, 2001, pp. 642–651.
[9] K. Chen, A constant factor approximation algorithm for k-median clustering with outliers, in: Proc. of the 19th ACM Symposium on Discrete Algorithms, 2008, pp. 826–835.

[10] R. Cole, L.-A. Gottlieb, Searching dynamic point sets in spaces with bounded doubling dimension, in: Proc. of the 38th ACM Symposium on Theory of Computing, 2006, pp. 574–583.
[11] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, MIT Press, 2001.
[12] J. Cuena, J. Hernandez, M. Molina, Knowledge-based models for adaptive traffic management systems, Transportation Research Part C: Emerging Technologies 3 (5) (October 1995) 311–337.
[13] G.D. da Fonseca, D.M. Mount, Approximate range searching: The absolute model, Computational Geometry: Theory and Applications 43 (2010) 434–444.
[14] B. Degener, J. Gehweiler, C. Lammersen, The kinetic facility location problem, Algorithmica (2008), doi:10.1007/s00453-008-9250-7.
[15] E.D. Demaine, F.V. Fomin, M. Hajiaghayi, D.M. Thilikos, Fixed-parameter algorithms for the (k, r)-center in planar graphs and map graphs, ACM Transactions on Algorithms 1 (1) (July 2005) 33–47.
[16] Y. Deng, B.S. Manjunath, NeTra-V: Toward an object-based video representation, IEEE Trans. on Circuits and Systems for Video Technology 8 (5) (1998) 616–627.
[17] T. Feder, D. Greene, Optimal algorithms for approximate clustering, in: Proc. of the 20th ACM Symposium on Theory of Computing, 1988, pp. 434–444.
[18] J. Gao, L.J. Guibas, A. Nguyen, Deformable spanners and applications, Computational Geometry: Theory and Applications 35 (1) (2006) 2–19.
[19] R. Gillmann, Accuracy assessment of traffic monitoring devices vehicle by vehicle, Transportation Research Record: Journal of the Transportation Research Board 1945 (2006) 56–60.
[20] A.R. Golding, Automobile navigation system with dynamic traffic data, U.S. Patent 5933100, filed December 27, 1995, and issued August 3, 1999.
[21] T. Gonzalez, Clustering to minimize the maximum intercluster distance, Theoretical Computer Science 38 (1985) 293–306.
[22] L.-A. Gottlieb, L. Roditty, An optimal dynamic spanner for doubling metric spaces, in: Proc. of the 16th European Symposium on Algorithms, 2008, pp. 478–489.
[23] A.P. Gribbon, Field test of nonintrusive traffic detection technologies, Math. Comput. Modelling 27 (9–11) (1998) 349–352.
[24] L. Guibas, Kinetic data structures, in: D. Mehta, S. Sahni (Eds.), Handbook of Data Structures and Applications, Chapman and Hall/CRC, 2004, pp. 23-1–23-18.
[25] L.J. Guibas, Kinetic data structures: A state of the art report, in: Proc. of the Third Workshop on the Algorithmic Foundations of Robotics, 1998, pp. 191–209.
[26] P. Gupta, R. Janardan, M. Smid, Fast algorithms for collision and proximity problems involving moving geometric objects, Computational Geometry: Theory and Applications 6 (1996) 371–391.
[27] S. Har-Peled, Clustering motion, Discrete & Computational Geometry 31 (4) (2004) 545–565.
[28] D.S. Hochbaum (Ed.), Approximation Algorithms for NP-Hard Problems, PWS Publishing Co., Boston, 1995.
[29] D.S. Hochbaum, D.B. Shmoys, A best possible heuristic for the k-center problem, Mathematics of Operations Research 10 (2) (1985) 180–184.
[30] O. Kariv, S. Hakimi, An algorithmic approach to network location problems. Part 1: The p-centers, SIAM Journal on Appl. Math. 37 (1979) 513–538.
[31] S.-W. Kim, Y. Eun, H. Kim, J. Ko, W.-J. Jung, Y.K. Choi, Y.G. Cho, D.-I. Cho, Performance comparison of loop/piezo and ultrasonic sensor-based traffic detection systems for collecting individual vehicle information, in: Proc. of 6th World Cong. on Intelligent Transport Systems, 1998 (CD-ROM).
[32] R. Krauthgamer, J.R. Lee, Navigating nets: Simple algorithms for proximity search, in: Proc. of the 15th ACM Symposium on Discrete Algorithms, 2004, pp. 798–807.
[33] R.M. McCutchen, S. Khuller, Streaming algorithms for k-center clustering with outliers and with anonymity, in: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, in: Lecture Notes in Computer Science, vol. 5171, 2008, pp. 165–178.
[34] J. Plesnik, On the computational complexity of centers locating in a graph, Applications of Mathematics 25 (6) (1980) 445–452.
[35] J. Plesnik, A heuristic for the p-center problem in graphs, Discrete Applied Math. 17 (3) (1987) 263–268.
[36] P.J. Rousseeuw, A. Leroy, Robust Regression and Outlier Detection, Wiley, New York, 1987.
[37] N. Saunier, T. Sayed, Automated analysis of road safety with video data, Transportation Research Record: Journal of the Transportation Research Board 2019 (2007) 57–64.
[38] E. Schomer, C. Theil, Efficient collision detection for moving polyhedra, in: Proc. of the 11th ACM Symposium on Computational Geometry, 1995, pp. 51–60.
[39] E. Schomer, C. Theil, Subquadratic algorithms for the general collision detection problem, in: Abstracts 12th European Workshop Comput. Geom., 1996, pp. 95–101.
[40] D. Wang, Unsupervised video segmentation based on watersheds and temporal tracking, IEEE Trans. on Circuits and Systems for Video Technology 8 (5) (1998) 539–546.
[41] Z. Wang, S. Elbaum, D.S. Rosenblum, Automated generation of context-aware tests, in: Proc. of the 29th International Conference on Software Engineering, 2007, pp. 406–415.