Dynamic L-Budget Clustering of Curves

Kevin Buchin
Faculty of Computer Science, TU Dortmund University, Germany

Maike Buchin
Faculty of Computer Science, Ruhr University Bochum, Germany

Joachim Gudmundsson
Faculty of Engineering, The University of Sydney, Australia

Lukas Plätz
Faculty of Computer Science, Ruhr University Bochum, Germany

Lea Thiel
Faculty of Computer Science, Ruhr University Bochum, Germany

Sampson Wong
Department of Computer Science, University of Copenhagen, Denmark

Abstract

A key goal of clustering is data reduction. In center-based clustering of complex objects therefore not only the number of clusters but also the complexity of the centers plays a crucial role. We propose L-Budget Clustering as unifying perspective on this task, optimizing the clustering under the constraint that the summed complexity of all centers is at most L. We present algorithms for clustering planar curves under the Fréchet distance, but note that our algorithms more generally apply to objects in metric spaces if a notion of simplification of objects is applicable. A scenario in which data reduction is of particular importance is when the space is limited. Our main result is an efficient $(8+\varepsilon)$-approximation algorithm with a $(1+\varepsilon)$-resource augmentation that maintains an L-budget clustering under insertion of curves using only $O(L\varepsilon^{-1})$ space and $O^*(L^3 \log L + L^2 \log(r^*/r_0))$ time where O^* hides factors of ε^{-1}.

2012 ACM Subject Classification Theory of computation → Facility location and clustering

Keywords and phrases clustering, streaming algorithm, polygonal curves, Fréchet distance, storage efficiency, simplification, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.SWAT.2024.18

Supplementary Material *Software*: `https://github.com/NathenMat/SWAT24` [7]
archived at `swh:1:dir:c28806ab33cb4a564e1d492efd322b1dc32f7245`

Funding *Lukas Plätz*: The work was supported by the PhD School "SecHuman – Security for Humans in Cyberspace" by the federal state of NRW.

1 Introduction

Clustering is a key technique for reducing dataset size in big data and serves as a fundamental analysis task. The goal is to keep data characteristics as close as possible to the original while limiting the size and complexity. In particular, when dealing with very large data, such as live traffic data, this needs to be compressed regularly. This necessarily leads to an approximation of the dataset, and thus we always pay an approximation factor in each later analysis. This tradeoff between space-efficient data storage and accuracy poses a challenge for long-term analysis. To address this, we introduce a clustering strategy that approximately bounds the clustering error within our storage constraints.

If the objects to be clustered are simply points, k-center clustering directly provides a compact representation, since it is sufficient to store k points as cluster centers Gonzalez [13] and Hochbaum and Shmoys [15] give 2-approximation algorithms for k-center clustering,

which is optimal assuming $P \neq NP$. These algorithms, though, require access to all points at once. Charikar et al. [12] present a dynamic k-center clustering algorithm for point sets, where points can be added to the clustering. Their doubling algorithm allows them to process the points one at a time, which gives an 8-approximation and can run using $O(k)$ space. This result was improved by McCutchen and Khuller [18] to a $(2+\varepsilon)$-approximation. A technique they use to achieve this is to run multiple instances with different radii to remove a factor from the approximation. This increases the runtime and memory usage by $O(\varepsilon^{-1} \log \varepsilon^{-1})$.

These techniques can only approximate storage requirements when all objects possess identical, small complexity. This assumption, however, may oversimplify the reality and prove too substantial for various types of data, such as libraries of images, movement-tracking trajectories, and networks. Even for seemingly simple objects like real numbers, their digital representation requires simplification, such as floating-point precision. Moreover, by adopting a simplified approach to clustering, they inadvertently constrain their options. Typically, this results in a binary decision: deleting or keeping the object as it is. In contrast, when we consider complex objects, they often bring a notion of simplifications.

In this work, we focus on polygonal curves as these occur naturally in many settings, and the literature provides a solid foundation of clustering, simplification, and similarity computation. They also show all the interesting features of complex objects. Nevertheless, our present techniques apply to a wider field of objects. The specific setting only influences the runtime analysis, and our clustering results hold in a more general setting. When we cluster curves, given some accuracy, we see that some may share a complicated center path, and others are similar in their position and have only a simple center path.

One option to deal with this setting is the (k,l)-*clustering problem*. It gives a clustering of complex objects by simplifying them to a uniform size of l. This saves the upper bound on the storage occupied by the clustering. Curves can be found in many applications and have different similarity metrics. We are interested in the Fréchet distance, which respects the traversal of a curve (cf. Hausdorff) but not its sampling (cf. dynamic-time-warping). For curves utilizing the Fréchet distance as the similarity metric, the centers are selected from the entire domain of Fréchet curves[1]. We choose the center curves freely i.e. under the unrestricted simplification setting. The selection process aims to minimize the maximum cluster radius. Buchin et al. [8] proposed a 3-approximation for this setting. Also for the setting of (k,l)-median clustering of curves, where not the maximum radius but the summed radii are considered, approximation algorithms have been developed [6, 10].

However, (k,l)-clustering may yield an imbalanced solution to the original problem, where some centers are inherently more complex than others. Additionally, determining the appropriate number of clusters in advance poses a challenge, often addressed by computing clustering for various values of k. Restricting the number of clusters can limit the full benefit of simplifications or the identification of cluster centers that genuinely represent the underlying data. Pre-determination of a suitable l may also be challenging and may depend on other considerations, such as the readability of centers.

We introduce L-*budget clustering* as a solution in this setting. The L-budget clustering problem is a center clustering problem with a complexity associated with each center. In a solution to it, the sum over the complexity of the centers is at most L and minimizes the maximum radius of the clusters. See Figure 1 for a small example.

[1] To establish the Fréchet Distance as a metric, we consider the equivalence classes of curves sharing the same trace, using the quotient space.

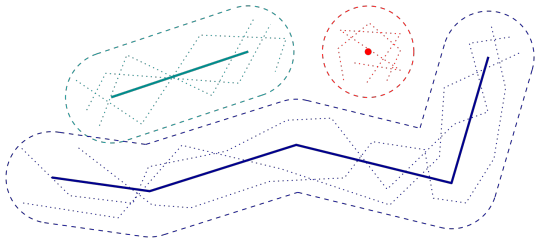

Figure 1 Small example of an L-budget clustering consisting of three clusters for $L = 8$.

Hence, we do not assume to know any meta parameters, nor do our guarantees rely on them, such as the number of clusters in our dataset or the objects' uniform complexity. Instead, we restrict ourselves to claims in the resource we are willing to spend: memory space. The strict size constraint allows us to optimize the representativeness using the given space. We assume that each input curve has complexity L. This can be achieved if necessary with a streaming simplification algorithm as described in Remark 9. To continuously build and analyze the data, we present a 1-pass streaming $(8 + \varepsilon)$-approximation algorithm for L-budget clustering of curves with computing space in the output size and a $(1 + \varepsilon)$-resource augmentation. That means it only needs to see the data once and does not require extra computation space, allowing us to handle large datasets. The approximation factor can be decreased to $(2 + \varepsilon)$ but with an increase of $O(\varepsilon^{-1})$ in runtime and space requirements.

Overview. In Section 2, we summarize the simplification problems for curves and the current techniques to solve them. Here, we present an algorithm for a $(1 + \varepsilon)$-approximation to find the minimum complex simplification with linear space. In Section 3, we present the static L-budget clustering problem and continue in Section 4 with the dynamic (k, l)-clustering. In Section 5, we then handle the dynamic L-budget clustering. We end with a short experimental evaluation in Section 6.

2 Simplification

Simplifying a curve is a natural and well-studied problem. It asks to reduce the complexity of a curve while keeping it as similar as possible to the original curve. For the similarity, different distance measures can be used. We will use the popular Fréchet distance $\mathbf{d}_F$ [4]. We denote the complexity of a curve z with $\mathbf{cplx}(z)$, which counts the number of points.

Simplifying a curve is then a bi-criterion optimization problem of the size and distance of the simplified curve (to the original curve). So, the literature discusses two subproblems. With min–#, we denote the simplification problem where we start with an upper bound on the Fréchet distance and minimize the complexity l of the simplification. The other variant is min–$\mathbf{d}$[2], where we start with an upper bound ℓ on the complexity and want to minimize the Fréchet distance $\mathbf{d}_F$. We are interested in the global unrestricted setting, the least-restricted simplification setting, to guarantee the best possible clustering. This allows us to get the least possible error given the size of the simplification, which in turn allows a better representation of the curves with the same budget. Global means that the Fréchet distance between the simplification and the curve is minimized [20]. This is in contrast to the local setting where the curve is partitioned. Then, the Fréchet distance is calculated

[2] We changed min–ε to min–$\mathbf{d}$ because it conflicts with the ε from full approximation schemas.

between the parts and its simplification, and the maximum over this is minimized [14]. In particular in earlier research on simplification, only curves with vertices of the original were considered as simplification [16]. This is the vertex-restricted case. Later, this was loosened to the unrestricted setting where the vertices can be from the whole metric space [14, 3]. See van de Kerkhof et al. [20] for a more detailed introduction.

We summarize relevant results on simplification in Table 1 and convert them to unrestricted global simplifications with Agarwal et al. [3]. This table contains offline and online algorithms which have different analysis settings. Offline algorithms are commonly analyzed with worst-case analysis and online algorithms with competitive ratios. However, simplification is a particularly hard setting and sometimes requires resource augmentation[3] [1, 20]. To unify the notation, we call the competitive ratio of the online algorithm just an approximation ratio but compare it to an offline algorithm that optimizes each instance individually.

Table 1 Results on Curve Simplification. In the min–**d** setting, first is the approximation ratio and then resource augmentation. In the min–# setting, it is reversed. The O^* hides terms in ε.

Authors and Paper	Setting	$(\mathbf{d}_F, \mathbf{cplx})$	Runtime	Space
Guibas et al. [14]	min–#, global, $\mathbb{R}^2$	$(1, 1)$	$O(m^2 \log^2 m)$	$O(m)$
Agarwal et al. [3]	min–#, global	$(1, 8)$	$O(m \log m)$	$O(l)$
van de Kerkhof et al. [20]	min–#, global	$(1+\varepsilon, 2)$	$O^*(m^2 \log m \log\log m)$	$O^*(m)$
Abam et al. [1]	min–**d**, global	$(16\sqrt{2}+\varepsilon, 2)$	$O^*(\ell)$	$O^*(\ell^2)$
Buchin et al. [8]	min–**d**, global	$(4, 1)$	$O(m^2 \ell \log m)$	$O(m)$
this paper Thm. 3	min–**d**, global, $\mathbb{R}^2$	$(1+\varepsilon, 1)$	$O^*((\log m + \ell) m^2 \log m)$	$O(m)$

One can change the simplification criterion of an algorithm by applying a binary search, as described by Chan and Chin [11].

For our clustering algorithms in Section 3, we need to find solutions to the min–# problem. We will call the algorithm that does that $\mathbf{S}_{\#}$. Guibas et al. [14] gave in theorem 14 with definition 4 an algorithm to solve this for planar curves in $O(m^2 \log^2 m)$ time.

We are interested in the min–**d**-simplification problem in the next sections, which we describe as finding a curve's best ℓ-simplification. This will be used in the later clustering algorithms. To prevent approximation factors in the space as much as possible, as this may be a hard constraint, we want an algorithm that has an competitive ratio of 1. For this we use a known algorithm by Imai-Iri, which was previously analysed and used by Buchin et al. [8]. Here, we briefly show how to get the $O(m)$ space and improve the approximation factor in Fréchet distance for planar curves using the disk stabber from Guibas et al. [14].

Kreveld et al. [21] gave an exact algorithm for min–#-simplification with dynamic programming in $O(m^2)$ space [4]. But this needs too much space for our application.

Our goal is to get a constant factor approximation in $O(m)$ space and improve it with a binary search to a $(1+\varepsilon)$-approximation. In the constrained space, we implicitly build the shortcut graph from Imai and Iri [16] and compute the best simplification given the length with a dynamic program. So, we compute the distance from the current edge to the sub-curve with the Fréchet distance algorithm from Alt and Godau [4]. The sub-curve can have at most length m, and the algorithm of Alt and Godau can run in $O(m)$ space. We compute the minimum Fréchet distance r necessary to reach a node i with an l-simplification

[3] Resource augmentation compares offline against online algorithms with more resources.
[4] In the look up table we only consider $k-1$ and can delete after a full iteration.

with the Algorithm 1 (Apx. Simplification). We save these values in a table $z[i,l]$. The solution to our problem is the value of $z[m,\ell]$, where m is the length of the input curve. To compute a $z[i,l]$, we need to compute the maximum of the previous $z[j,l-1]$ and the Fréchet distance between the sub curve $c[j,i]$ and the shortcut $\overline{c[j,i]}$, which we notate as $\mathbf{d}_F(\overline{c[j,i]}, c[j,i])$). We then can compute the minimum over all j. As we only need the last iteration over the length of the simplification, we only need space in the size of the number of points in the curve, which is $O(m)$.

Algorithm 1 Apx. Simplification.

Data: polygonal curve z, complexity of the simplification $\ell \in \mathbb{N}$
Result: subsequence of z of length at most ℓ with minimal Fréchet distance to z
for $l \leftarrow 1$ *to* ℓ **do**
  for $i \leftarrow 0$ *to* m **do**
    $z[i,l] \leftarrow \min(\max(\{z[j,l-1], \mathbf{d}_F(\overline{c[j,i]}, c[j,i])) \mid j < i\})$
  delete all $z[\cdot, l-1]$

▶ **Lemma 1** (By Imai and Iri [16]). *Algorithm 1 (Apx. Simplification) gives us an optimal vertex-restricted local Fréchet simplification.*

▶ Remark 2. Algorithm 1 (Apx. Simplification) gives a 4-approximation of the weak unrestricted simplification [3]. It has an $O(m^2 \ell \log m)$ runtime [8] and needs $O(m)$ space.

So, we have established an interval of constant size where the optimal value lies. We now do a binary search and decide with the algorithm from Guibas et at. [14] if an ℓ-simplification exists.

Algorithm 2 $(1+\varepsilon)$-Apx. Simplification.

Data: polygonal curve z, complexity of the simplification $\ell \in \mathbb{N}$
Result: subsequence of z of length at most ℓ with minimal Fréchet distance to z
$s \leftarrow Apx.Simplification(z, \ell)$
$h \leftarrow \mathbf{d}_F(s, z)$
$l \leftarrow h/4$
while $h - l > \varepsilon$ **do**
  $m \leftarrow (h+l)/2$
  if $\mathbf{len}(\mathbf{S}_\#(z,m)) \leq \ell$ **then** $h \leftarrow m$ **else** $l \leftarrow m$
return h

▶ **Theorem 3.** *Algorithm 2 ((1+ε)-Apx. Simplification) computes a $(1+\varepsilon)$-simplification of the* min–***d*** *without resource augmentation. The runtime is in $O((\log \varepsilon^{-1} \log m + \ell) m^2 \log m)$ and the space is in $O(m)$.*

Proof. The initial approximation needs $O(m^2 \ell \log m)$ runtime. To improve the approximation factor from 4 to $(1+\varepsilon)$, we need $\log(4\varepsilon^{-1})$ iterations of binary search. As the decision algorithm, we take [14] that needs $O(m^2 \log^2 m)$ time. This gives us the $O(\log \varepsilon^{-1} m^2 \log^2 m + m^2 \ell \log m)$ runtime. ◀

▶ Remark 4. We will later want to use the clustering algorithm for point sets from Charikar et al. [12]. It uses different radii r over its runtime. Storing the simplifications for each r will not be possible because of the space restriction of $O(L\varepsilon^{-1})$. Thus, we compute a *range of simplifications* that use $O(L\varepsilon^{-1})$ space and choose a simplification later when r increases.

The range of simplification are all simplifications of size $(1+\varepsilon)^{-i}L$ for all $i \in [-1, \log_{(1+\varepsilon)} L]$. The geometric series bounds the size of the range with $O(L\varepsilon^{-1})$. See Figure 2 for a small example.

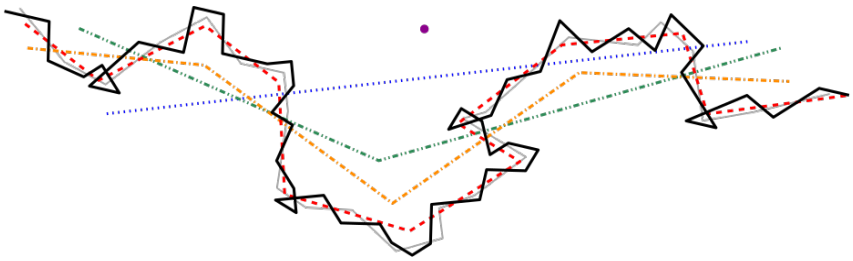

Figure 2 A non-optimal simplification range with factor 2. The colors and line styles indicate different stages of simplifications from full complexity (black line) down to 1 (purple point).

Algorithm 3 Simplification Range.

Data: polygonal curve z, base b, upper bound $\ell \in \mathbb{N}$
Result: sequence of simplifications s with complexity b^i and minimal Fréchet distance to z

```
s, l ← [ ], 1
while bℓ > l do
    s.append(S_#(z, l))
    l ← bl
return s
```

▶ **Lemma 5.** *Computing the range of simplifications with Algorithm 3 (Simplification Range) has $O^*(m^3 \log m)$ runtime.*

Proof. Summing up the runtimes of Algorithm 2 ((1+ε)-Apx. Simplification), we get

$$O\left(\sum_{i=0}^{\log_{(1+\varepsilon)} m} \left(\log \varepsilon^{-1} m^2 \log^2 m\right) + \sum_{i=0}^{\log_{(1+\varepsilon)} m} \left(m^2 (1+\varepsilon)^i \log m\right)\right).$$

The first sum simplifies to $O(\varepsilon^{-1} \log \varepsilon^{-1} m^2 \log^3 m)$. The second sum is a simple geometric series. This saves a $\log m$ factor and resolves to $O(\varepsilon^{-1} m^3 \log m)$ time. ◀

3 Static L-budget clustering

We consider the setting of clustering curves with a given L space restriction. That is, we want to find a clustering $\mathcal{C}$ of curves c from the domain of Fréchet curves with $\sum_{c \in \mathcal{C}} \mathbf{cplx}(c) \leq L$ such that the balls $\mathbf{B}_r(c)$ with center at c and radius r cover the set of input curves Z and r is minimal. A similar setting is the k-weighted center clustering discussed by Hochbaum and Shmoys [15].

This also is a bi-criterion problem and leads to the co-problem where we have a maximum radius r and want to know the clustering with the minimal sum over the complexity of the centers. The co-problem will be used in the proofs of our clustering algorithm.

Starting with the static setting of computing a space-efficient clustering, we present an algorithm for L-budget clustering. We begin with the description of Algorithm 4 (Decide L-Budget Clustering). Here, we are given a radius r and want to decide if an L-budget clustering of the curves with $3r$ exists or if no cover with radius r exists. We now test if a clustering with radius r fits our storage in the following way. Given our r, we compute the best curve simplification with the function $\mathbf{S}_{\#}$ for every curve and store them in a heap. Then, we draw simplifications of uncovered curves, starting with the curve that has the least verticies. For our simplification s, we compute the ball $\mathbf{B}_{3r}(s)$ and mark all curves as covered. The newly covered curves form a cluster with s as the center. We repeat the drawing until all curves are covered or exceed the budget L. If we succeed, we get a 3-approximation; otherwise, we know that r is smaller than the optimum radius r^* and retry with a larger r.

Algorithm 4 Decide L-Budget Clustering.

Data: set of curves Z, budget $L \in \mathbb{N}$, radius r
Result: decision $r^* \leq 3r$ or $r^* > r$
$C, l \leftarrow [\,], 0$
heap $\leftarrow$ build_heap($\{(\mathbf{cplx}(s), s, z) \mid z \in Z, s = \mathbf{S}_{\#}(z, r)\}$)
while $Z \neq \emptyset$ **do**
  $\ell, c, z \leftarrow$ heap.pop()
  if $z \notin Z$ **then** continue
  $l \leftarrow l + \ell$
  if $l > L$ **then return** $r^* > r$
  $C \leftarrow C \cup \{c\}$
  $Z \leftarrow Z \setminus \mathbf{B}_{3r}(c)$
return $r^* \leq 3r$

▶ **Theorem 6.** *Algorithm 4 (Decide L-Budget Clustering) decides $r^* \leq 3r$ or $r^* > r$ for the L-budget clustering problem.*

Proof. If the algorithm returns $r^* \leq 3r$, then it covers the curves, and the complexity of the centers is lower or equal to the budget. Therefore, $3r \geq r^*$.

If the algorithm returns $r^* > r$, we look at the following auxiliary clustering problem. We start with an r and want the least complex clustering $\mathcal{C}_r^*$. With it, we show that $\mathbf{cplx}(\mathcal{C}_r^*)$ is larger than the interim clustering computed by the algorithm. The complexity of the auxiliary optimal clustering proves that given the budget L, the optimal clustering has a radius larger than r. The interim clustering with radius r covered a set of curves with balls of radius $3r$ and had a complexity l over the budget L. We charge the complexity of the interim clustering onto the optimal solution $\mathcal{C}_r^*$ of the auxiliary clustering problem to show that there cannot be a solution with the given r. We start with charging the complexity l of the interim clustering to the centers with their respective complexity. Now, every center c has a curve z we used to generate it. We call z the *witness* of c. So, we charge the complexity of the center to its witness curve. The witness curve z has to be part of a cluster in the optimal clustering $\mathcal{C}_r^*$. So, we can forward the charge of the witness curve z to the center c^* of the optimal cluster.

Now, two properties play a key role. First, because we only considered simplification with an uncovered witness, the distance between witnesses must be at least $2r$. Thus, each witness is in a different optimal cluster, and each optimal center got charged at most once. Secondly, the optimal center c^* each got charged their complexity or less, as each of our centers c is

the minimal complexity curve in the r-ball $\mathbf{B}_r(z)$ around their witness z, which includes the optimal centers c^*. So, if we sum up the complexities of the charged optimal centers, we get that they are greater than the budget L. Hence, r was too small, and we get $r^* > r$. ◀

▶ **Theorem 7.** *Algorithm 4 (Decide L-Budget Clustering) with n curves of complexity m has runtime in $O((m \log^2 m + L)nm)$ where L is the complexity budget of the clustering.*

Proof. First, we compute all simplifications of Z for radius r. This takes $O(nm^2 \log^2 m)$ with the algorithm of Guibas et al. [14] in $\mathbb{R}^2$. Then we build a heap in $O(n)$ If we stop early because we reached the budget, we get that the summed complexity of the simplifications for which we computed the covering is $O(L)$. We also get that when we do not stop early and cover all curves. Therefore deciding which center covers which curve can be done naively in $O(Lnm)$ time. ◀

With the decider, we can search for the smallest r with $r^* \leq 3r$. Because we have no good upper bound, we start with an exponential search to find an upper bound. Here, we start with an initial guess r_0 and increase it by factors of 3. When we reach a cover of the curves, it also gives us a lower bound. With a binary search, we can find our smallest r up to an error of ε.

▶ **Theorem 8.** *For n curves of complexity m, the initial radius r_0, the optimal radius r^*, and L is the complexity of the clustering we can compute with Algorithm 4 (Decide L-Budget Clustering) in $O((m \log^2 m + L)nm(\log(r^*/r_0) + \log(\varepsilon^{-1})))$ time a $(3+\varepsilon)$-approximation.*

Proof. We invoke the decider $O(\log(r^*/r_0) + \log(\varepsilon^{-1}))$ many times. This gives the runtime. The correctness follows from the decider's correctness and the binary search's precision. ◀

We gained two insights from this section, which we will use later in the dynamic setting.

- We used a witness z to prove that the center has minimal complexity in a ball of radius r around it.
- The centers have a minimum distance between them to guarantee that no optimal center gets charged twice.

4 Dynamic (k, ℓ)-clustering

Next, we introduce the *dynamic setting*, which allows us to incrementally add curves to our clustering. Formally, in the dynamic setting, we get a sequence of curves. At each step, we have the previous clustering and a new curve. We want to compute a constant factor approximation to the optimal solution of the static problem over the subset of processed curves. In other words, we want an approximation to the static clustering problem but cannot see all curves simultaneously.

Because every cluster has its complexity budget in the (k, ℓ)-clustering problem, we have no tradeoff in the complexity between clusters. We assume that all input curves are of complexity ℓ in the dynamic setting.

▶ Remark 9. This can be achieved with various simplification algorithms. For inputs up to the size of $O(k\ell)$, we can get an ℓ-simplification using the algorithm of Buchin et al. [8]. They need $O(k^2\ell^3 \log(k\ell))$ time and $O(k\ell)$ space and return an 4-approximation in the Fréchet distance with a no resource augmentation in the complexity. Depending on your setting, this is a valid simplification algorithm for larger input sizes but will need more space. In many clustering applications, the number of objects is much larger than their complexity. Hence, the complexity of an object is typically not larger than the complexity

of the resulting clustering. However, if we want to process arbitrarily big input, Abam et al. [1] have a streaming algorithm for l-simplification with a competitive ratio of 2 and an $4\sqrt{2}+\varepsilon$-approximation in the Fréchet distance in $O(\ell^2/\sqrt{\varepsilon})$ space. But, of course, both simplification algorithms introduce an approximation factor in the complexity and the radius.

Assuming uniform complexity ℓ of the input curves allows us to use the "doubling algorithm" of Charikar et al. [12] in our setting of curve clustering. It computes a lower bound r and provides an upper bound αr of the optimal clustering radius r^*. McCutchen and Khuller [18] extended their algorithm to the "scaling algorithm" by allowing different approximation factors. They also devised a trick to improve the approximation factor. The "scaling algorithm" increases r multiplicatively by α only when there are more than k clusters. When r increases, the algorithm merges all clusters with a distance less than $2\alpha r$. Until the number of clusters is more than k, it tries to insert points into the existing clusters. Inserting a point into an existing cluster is possible if the distance between the center and the point is less than $\eta := 2\alpha^2/(\alpha-1)r$. So, when we merge clusters, we have to check that we do not immediately break the radius of the clusters. These constraints give us this inequality to be satisfied: $2\alpha r + \eta r \leq \eta\alpha r$. This simplifies to $1 < \alpha$. Charikar et al. also showed that the returned r is always smaller than the optimal radius r^*.

For the merging step, there are two variants. The first variant is to compute the threshold graph with a heap of all edges (using edge lengths as priority). This is the runtime-efficient implementation because there can be $O(k^2)$ many edges between the clusters. When we have a new cluster, we compute the distance of the center to all other centers and put an edge with the length as the priority into the heap. To build the t-threshold graph, we pop all edges from the heap with a length below or equal to t. The other variant computes the distance just in time. This only needs $O(k)$ space as each cluster can have that many edges but also requires re-computation of the distance each time. The algorithm would also work if we only had the nearest neighbor or range queries. There is some literature on this topic for the Fréchet distance [2, 5, 19], but we did not consider it because the bottleneck in the runtime analysis is the simplification.

We combine these properties and define a cluster as *valid* if and only if its radius is smaller than ηr. And clustering is valid if and only if all clusters are valid, $r \leq r^*$, the centers cover the pointset, the centers have distance $r\alpha$, and the clustering has at most k clusters.

We summarize the algorithm in pseudo-code as Algorithm 5 ((k,ℓ)-Scaling). We used the Python keyword *yield*. It behaves like a return and gives an output but allows the function to continue in a subsequent call at the point where it last yielded a result. Combined with *next*, this allows a nice notation of iteration over sequences or generators.

We start the runtime analysis with an update step.

▶ **Lemma 10.** *Algorithm 5 ((k,ℓ)-Scaling) computes an update step, excluding the insertions of the curves, in $O(k\ell^2 \log \ell + k \log k)$ amortized time.*

Proof. The algorithm maintains a heap of the edges in order of their length. The graph can have almost $O(k^2)$ many edges. A new cluster introduces an edge to all the other clusters. To compute the Fréchet distances, we need $O(k\ell^2 \log \ell)$ time. We charge each edge with $(\ell^2 \log \ell + \log k)$ at the insertion into the heap. When the threshold t increases, we need all edges below t to build the threshold graph. Deletion of i edges from a heap takes $O(i \log k)$ time. However, we paid for the deletion at the insertion to get $O(k\ell^2 \log \ell + k \log k)$ amortized time. ◀

We can then bound the runtime of the whole Algorithm 5 ((k,ℓ)-Scaling).

Algorithm 5 (k, ℓ)-Scaling.

Data: sequence of curves Z, number of clusters $k \in \mathbb{N}$, complexity of center $\ell \in \mathbb{N}$, approximation factor α
Result: sequence of valid clustering $\mathcal{C}$ with their respected radius r
`/* initialization and small value treatment */`
$\mathcal{C} \leftarrow \emptyset$
for $i \in [k]$ **do**
  $z \leftarrow \textbf{next}(Z)$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{(z, \{z\})\}$
  yield $\mathcal{C}, 0$
$2\alpha r \leftarrow$ minimal distance of any pair in the first $k+1$ curves
`/* the core of the algorithm */`
while $z \in Z$ **do**
  $z \leftarrow \textbf{next}(Z)$
  if $\boldsymbol{B}_{\eta r}(z) \cap \mathcal{C} =$ **then** $\mathcal{C} \leftarrow \mathcal{C} \cup \{z\}$
  `/* merge step */`
  while $|\mathcal{C}| > k$ **do**
    $r \leftarrow \alpha r$
    merge clusters with the $2\alpha r$-threshold graph
  yield $\mathcal{C}, r$ `// generator notation from python`

▶ **Theorem 11.** *Algorithm 5 ((k, ℓ)-Scaling) yields an 8-approximation and computes a clustering of n curves with cluster radius r^* in $O((k\ell^2 \log \ell + k \log k) \log(r^*/r_0) + kn\ell^2)$ amortized time.*

Proof. The correctness follows from McCutchen and Khuller [18] and the performance ratio is described by $\eta r / r^* \leq \eta$ and $\eta := 2\alpha^2/(\alpha - 1)$. This is at its smallest at $\alpha = 2$.

We get the number of update steps with $\log(r^*/r_0)$. This only needs to be multiplied by the runtime of the update step. After adding the number of decisions of the Fréchet distance, we need to check if a curve fits into an existing cluster. We get a runtime of $O((k\ell^2 \log \ell + k \log k) \log(r^*/r_0) + knl^2)$. ◀

We can improve the approximation factor with the trick from McCutchen and Khuller [18]. It uses multiple instances of the "scaling algorithm" with different start values. This leads to the approximation factor of $(2 + \varepsilon)$ and the runtime and space increases by $O(\varepsilon^{-1} \log \varepsilon^{-1})$.

5 Dynamic L-budget clustering

Now, we consider clustering with a fixed budget in the dynamic setting. We assume each input curve has at most complexity L, i. e. is an L-simplification if necessary. We provide an $\eta(1 + \varepsilon)$-approximation of dynamic L-budget clustering when using $(1 + \varepsilon)L$ as budget. In L-budget clustering, we can trade between cluster complexity and the number of clusters. The novel aspect of our setting is that we have differently complex centers, and the complexity of a simplification can change. Simplifying gives a new way to lower the complexity of a cluster by replacing the current center with a simplification of it. We show that there is a natural moment when we will simplify and when we will merge clusters.

For the construction, we need a witness of the minimal complexity for each cluster center and each r. We always have to guarantee a $(1+\varepsilon)$-approximation of the optimal simplification in the ball $\mathbf{B}_r$. However, we cannot compute these simplifications later and must work with the range of stored simplifications. This means we will have such simplification only up to a resource augmentation of $(1+\varepsilon)$. We build on the work of Charikar et al. [12] and McCutchen and Khuller [18], who introduce the approximation factor η. A clustering is valid if the maximum radius is smaller or equal to $\eta(1+\varepsilon)r^*$ and the sum of the complexity of the centers is smaller or equal to $(1+\varepsilon)L$. We will show that our incremental algorithm adds a curve and constructs a new valid clustering given a previously valid clustering.

In Algorithm 6 (Initialization), we get our first curve z and compute all the $(1+\varepsilon)$-simplifications with algorithm $\mathbf{S_d}$ (see Section 2). Our simplification algorithm gives the best guarantees, needing a resource augmentation of only $(1+\varepsilon)$ but works only for curves in 2D. We represent a cluster with a set of tuples of the center, the witness's influence, and the cluster radius's upper bound for each simplification. We define $\mathbf{d}_F(c_{-1}, z) := \infty$.

Algorithm 6 Initialization.

Data: polygonal curve z, budget $L \in \mathbb{N}$
Result: valid clustering $\mathcal{C}$ with its respected radius r
`/* (center, witness influence, upper bound on cluster radius) */`
$\mathcal{C} \leftarrow \{((c_i, \mathbf{d}_F(c_{i-1}, z), \mathbf{d}_F(c_i, z)) \mid c_i \in SimplificationRange(z, (1+\varepsilon), L)\}$
$r, l \leftarrow \mathbf{d}_F(c_0, z), L$
return $\mathcal{C}, r$

▶ **Lemma 12.** *Algorithm 6 (Initialization) gives a valid clustering, the runtime is in* $O(\varepsilon^{-1}(\log \varepsilon^{-1} \log^2 L + L)L^2 \log L)$ *and the space is in* $O((1+\varepsilon)L)$.

Proof. We get the complexity of the centers by construction. For the simplification algorithm, we already showed that this produces a $(1+\varepsilon)$-simplification to the optimal simplification. Because the distance between the curve and the simplification monotonically decreases in the number of nodes, we get that the highest complex simplification has to be as good as the optimal simplification. Thus, we get that $r \leq r^*(1+\varepsilon)$. From this, it then follows that the clustering is valid.

The runtime is dominated by the runtime of Algorithm 3 (Simplification Range). ◀

So, we can assume that we have a valid clustering for the main algorithm and show that we maintain it when we add a new curve w.

Algorithm 7 L-Budget Main.

Data: sequence of curves Z, budget $L \in \mathbb{N}$
Result: sequence of valid clustering $\mathcal{C}$ with their respected radius r
$z \leftarrow$ **next**(Z)
$\mathcal{C}, r \leftarrow$ Initialization(z, L)
yield $\mathcal{C}, r$
for $z \in Z$ **do**
  $\mathcal{C} \leftarrow$ Insertion$(z, \mathcal{C}, \eta r)$
  $\mathcal{C}, r \leftarrow$ Make Clustering Valid$(\mathcal{C}, L, r)$
  yield $\mathcal{C}, r$

The structure of our Algorithm 7 (L-Budget Main) is that of the algorithm from Charikar et al. [12], but we changed the while loop to be controlled by the complexity of the clustering. The same two cases can happen.

In the first case, we can insert with Algorithm 8 (Insertion) the curve z without creating a new cluster. Then, l and r do not change.

Algorithm 8 Insertion.

Data: polygonal curve z, clustering $\mathcal{C}$, radius r, budget $L \in \mathbb{N}$
Result: updated clustering $\mathcal{C}$ containing z
`/* inserting curve z if possible */`
for $C \in \mathcal{C}$ **do**
  if $z \in \boldsymbol{B}_{\eta r}(C_{center})$ **then**
    $C \leftarrow (c_i, w_i, \max(r_i, \mathbf{d}_F(c_i, z)))$
    return $\mathcal{C}$
`/* z has distance > r to all other cluster centers. */`
$C \leftarrow \{(c_i, \mathbf{d}_F(c_i, z), \mathbf{d}_F(c_i, z)) \,|\, c_i \in SimplificationRange(z, (1+\varepsilon), L)\}$
`/* reduce the complexity of the centers and guarantee the witness */`
$C \leftarrow ShortenCenterList(C, r)$
$\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$
return $\mathcal{C}$

When we insert z into a cluster, we compute the distance to all simplifications of the center and update the upper bound on the maximum radius. In the second case, we could not find a cluster for z, so it has to introduce a new one. For this, we do the same as in the initialization and remove all the centers with an upper bound lower than r up to the lowest complex one. This is done with Algorithm 9 (Shorten Center List). There has to be at least one simplification with the upper bound lower or equal to r because we assume the curve has complexity at most L and so, the curve itself can be a center. We also computed the radius r' of the by factor $(1+\varepsilon)$ more complex simplification. So, we can guarantee that the simplification is the best for any radius smaller than r' with the resource augmentation of $(1+\varepsilon)$.

Algorithm 9 Shorten Center List.

Data: center set C, radius r
Result: reduced center set C with minimal complexity for the given r
$idx \leftarrow \max_{\{(c_i, w_i, r_i) := C[i], w_h \leq r\}} h$
return $C[idx :]$

▶ **Lemma 13.** *Algorithm 9 (Shorten Center List) reduces the list of centers to the minimal complexity given r and keeps the cluster valid, runs in place and in $O(\log \log L)$ time.*

Proof. The center list contains the optimal centers for the opening curve z up to a factor of $(1+\varepsilon)$ in complexity. Before the shortening, the curves in the cluster had at most ηr_{old} distance from the old center. The old center was in r_{old} of z, and z is in r_{new} of the new center. The radius r_{new} is at least α times the old r_{old}. Therefore, the distance to the new center is at most $(\eta + \alpha + 1) r_{old}$. This, of course, needs to be less than $\eta \alpha r_{old}$, which is true for all our $\alpha > 1$. This proves that the new center keeps the cluster valid. The next center would be farther than r to z, invalidating z as a witness.

The values of w_i have to be monotonically increasing. The list length is $O(\log L)$, and binary search inserts another log. ◀

We use Algorithm 10 (Make Clustering Valid) to deal with too complex clusterings. It starts with increasing the distance r, which allows us to simplify the cluster centers further. We finish if the total complexity is at most the budget, and the pairwise center distances are at least distance 2α. Otherwise, we merge clusters with the $2\alpha r$-threshold graph.

Algorithm 10 Make Clustering Valid.

Data: clustering $\mathcal{C}$, budget $L \in \mathbb{N}$, radius r
Result: valid clustering $\mathcal{C}$ with its respected radius r
while $l > (1+\varepsilon)L$ **do**
  $r \leftarrow \alpha r$
  `/* simplifying the centers and changing the witness */`
  for $C \in \mathcal{C}$ **do**
    Shorten Center List(C, r)
  `/* reducing the numbers of clusters */`
  merge clusters with the $2\alpha r$-threshold graph
  $l \leftarrow \mathbf{cplx}(\mathcal{C})$
return $\mathcal{C}$, r

▶ **Theorem 14.** *After Algorithm 10 (Make Clustering Valid), we have a valid clustering for the curves considered, and an outer loop needs $O(L^2)$ time.*

Proof. The algorithm covers the curves, and the complexity of the centers is, at most, the budget by construction. The maximum radius is ηr implying $r^* \leq \eta r$. We now show that $r \leq r^*$. See Figure 3 for a visualization of the flow of charge and the inter-cluster distance.

Before the last loop in the merge loop, we covered the curves with complexity over the budget and radius ηr_{old}. We will charge this onto the optimal solution to show that there cannot be a solution with the given r_{old}. We start with charging the complexity of the solution to the centers with their respective complexity. Now, every center c_i has a witness w_i who is part of its cluster and proves the minimality of the center up to the factor of $(1+\varepsilon)$. So, we charge the center's complexity times the factor of $(1+\varepsilon)$ to its witness and then to the optimal center c_i^*. We cannot charge two curves in the same optimal cluster as we merge clusters within $2\alpha r_{old}$. Hence, each optimal center got charged at most once. However, the optimal centers each got charged their complexity times the factor of $(1+\varepsilon)$ or less, as each of our centers is the minimal complexity curve in the range of the witness up to the factor of $(1+\varepsilon)$. If we now sum up the charges on the optimal centers, we get a lower bound of $(1+\varepsilon)$ times the complexity of the optimal solution. But our initial complexity is greater than $(1+\varepsilon)$ times the budget L.

Because of the size constraint, we cannot maintain the threshold graph in memory, so we need to decide each edge, which takes $O(L^2)$ time by deciding the Fréchet distance between every cluster. ◀

▶ **Lemma 15.** *Algorithm 7 (L-Budget Main) gives a valid solution for each iteration.*

Proof. If we insert a curve into a cluster, everything stays the same, inheriting the validity. If we must introduce a new cluster, the merge loop implies the solution's validity. ◀

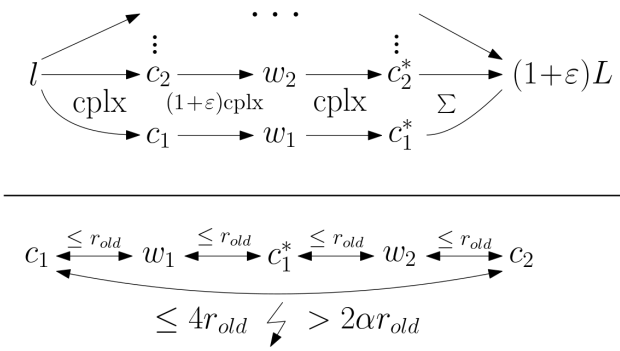

Figure 3 On top is an illustration of the flow of charge and below inter-cluster distance requirement.

▶ **Theorem 16.** *Algorithm 7 (L-Budget Main) gives an $(8+\varepsilon)$-approximation ratio with a $(1+\varepsilon)$-resource augmentation, needs $O^*(L^3 \log L + L^2 \log(r^*/r_0))$ time and $O(L\varepsilon^{-1})$ space with r_0 being the inital and r^* the optimal radius.*

Proof. The performance ratio of the clustering is $\eta r/r^* \leq \eta$ with $\eta := 2\alpha^2/(\alpha-1)$ which is minimal at $\alpha = 2$. Multiplying the simplification error gives us $(8+\varepsilon)$.

The worst-case path through the algorithm first tests every existing cluster to see if the new curve fits. This needs multiple decisions of Fréchet distances, which take $O(L^2)$ time. Introducing a new cluster and computing the range of simplifications takes $O(\varepsilon^{-1}(\log \varepsilon^{-1} \log^2 L + L)L^2 \log L)$ time. Each outer loop of Algorithm 10 (Make Clustering Valid) needs $O(L^2)$ time. Because we will need at most $\log(r^*/r_0)$ many of them, we get $O(\varepsilon^{-1}(\log \varepsilon^{-1} \log^2 L + L)L^2 \log L + L^2 \log(r^*/r_0))$ time.

The space requirements of computing the simplification is $O(L)$, and the size of the data structure for the clusters is in $O(l\varepsilon^{-1})$. The complexity only increases if a new cluster is added, but $l \in O(L)$ because curves have at most complexity L. ◀

The trick from McCutchen and Khuller [18] can be applied because they rely on α getting big, which coincides with our restriction to not overcharge the optimal clusters – leading to a $(2+\varepsilon)$-approximation but runtime and space increases by $O(\varepsilon^{-1})$.

Reintroducing the simplification and using the algorithm from Abam et al. [1] of the curve to fit the $O(L)$ space assumption adds the space requirement of $O(L^2)$, adds the $(16\sqrt{2}+\varepsilon)$-approximation factor, and increases the resource augmentation to $(2+\varepsilon)$.

6 Experiments

We conducted experiments to compare the results of L-budget clustering to (k,ℓ)-center clustering and their respective runtimes. We use Dennis Rohde's C++ implementation for Fréchet distance, simplification, and (k,ℓ)-clustering[5], in which we added a decider for the Fréchet distance and used it in the implementation of the simplification of Agarwal et al. [3]. As dataset, we used the trajectories of homing pigeons [17] also used by [9] to demonstrate (k,ℓ)-center clustering. We computed clusterings for varying budget L and varying number n of curves from the data set. We compare our results to the best $(k, L/k)$-clustering found in a linear search over all $k \in [L]$. Figure 4 displays one result of a (k,ℓ)-center and a L-budget clustering on this data. The curves contain some outliers and a central cluster detected by

[5] `https://github.com/derohde/Fred`

both clusterings. The central cluster in the middle is quite straight with minor deviations. The outliers to the north and west make quite a detour and hence need a complex center to be represented well. Both algorithms found 5 clusters but static L-budget with half the radius ($r = 0.0097$). For this inhomogeneity in cluster complexity, L-budget clustering was designed. The (k, ℓ)-center clustering, in contrast, overfits the middle cluster without improving the clustering radius, but also has too little complexity to represent the outliers.

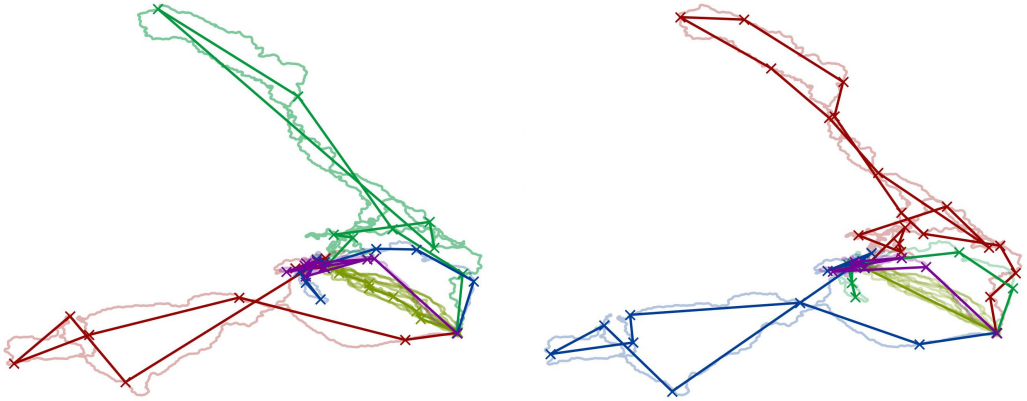

Figure 4 Example of a (k, ℓ)-center (left) and L-budget (right) clustering on a $n = 25$ curves using a budget of $L = 50$. Clusters are indicated by color and cluster centers have marked vertices.

Figure 5 shows the resulting runtimes.

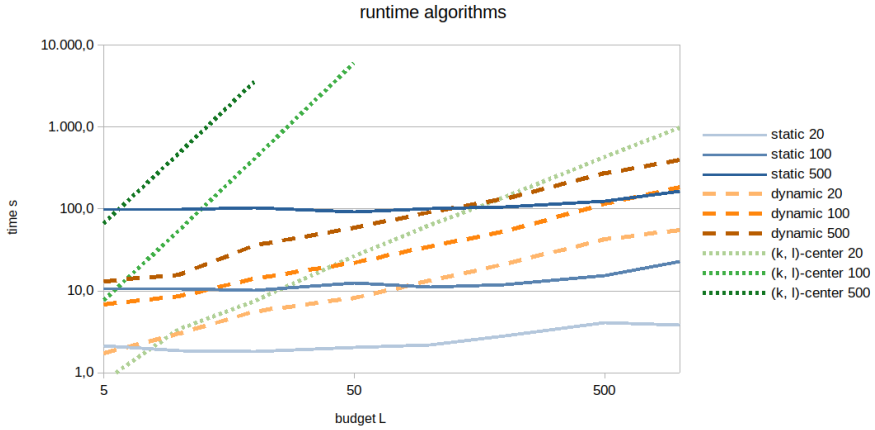

Figure 5 Shown are the runtimes of the algorithms in seconds. The runtime of some of the (k, l)-clustering experiments exceeded our time and did not finish.

The runtimes grow in L and n, seemingly slower than linear in L or n. This could be because the simplification algorithm finds long shortcuts quickly, and the clustering reduces most Fréchet distance decisions to clear-cut cases where at least one curve is of low complexity. For all interesting L and n, we also see that both implementations of L-budget clustering are much faster than using (k, ℓ)-center clustering. This is also the case for larger n or L if we compute only one instance of (k, ℓ) clustering.

To compare the results, we divide the radius found by the L-budget clustering by the best radius found by the (k, l)-center clustering in Figure 6.

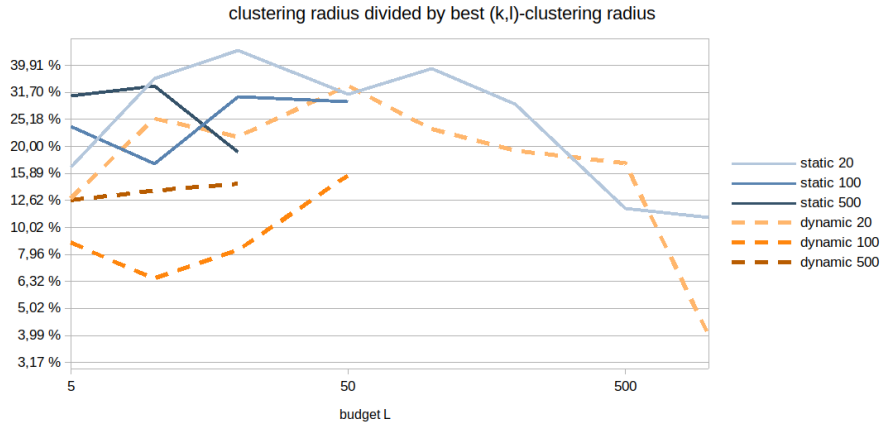

Figure 6 Shown is the radius size in percent against the best possible (k, ℓ)-center clustering. Some curves stop early because of the missing reference clustering.

These experiments suggest that L-budget clustering finds clusterings with significantly smaller clustering radius than a comparable (k, ℓ)-center clustering instance. The faster runtime enables the clustering of much larger instances.

7 Conclusion

We have addressed center clustering problems of complex objects under space constraints. We introduced the L-budget clustering problem to handle the issue and presented a 3-approximation for the static setting. We continued with the dynamic (k, ℓ)-clustering and then considered the space-restricted streaming setting. For the streaming setting, we presented an $(8 + \varepsilon)$-approximation algorithm with a $(1 + \varepsilon)$ resource augmentation over the budget L and gave a $(2 + \varepsilon)$-approximation with resource augmentation but with increased runtime and space requirements. We concluded with proof-of-concept experiments showing that the clustering is fast in practice and produces good results.

For L-budget clustering the objects to be clustered require a notion of simplification. While for curves streaming algorithms for simplification are known, it remains open is to find such an algorithm that works in $O(\ell)$ space. It would also be interesting to consider L-budget median clustering, where not the maximum but the summed radii are considered.

References

1 Mohammad Ali Abam, Mark de Berg, Peter Hachenberger, and Alireza Zarei. Streaming algorithms for line simplification. In *Proc. 23rd SoCG*, pages 175–183. ACM, 2007. doi:10.1145/1247069.1247103.

2 Peyman Afshani and Anne Driemel. On the complexity of range searching among curves. In *Proc. 2018 SODA*, pages 898–917, 2018. doi:10.1137/1.9781611975031.58.

3 Pankaj K Agarwal, Sariel Har-Peled, Nabil H Mustafa, and Yusu Wang. Near-linear time approximation algorithms for curve simplification. *Algorithmica*, 42:203–219, 2005.

4 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.*, 5:75–91, 1995.

5 Julian Baldus and Karl Bringmann. A fast implementation of near neighbors queries for Fréchet distance (GIS cup). In *Proc. 25th ACM SIGSPATIAL*, SIGSPATIAL '17, 2017. doi:10.1145/3139958.3140062.

6 Milutin Brankovic, Kevin Buchin, Koen Klaren, André Nusser, Aleksandr Popov, and Sampson Wong. (k, l)-medians clustering of trajectories using continuous dynamic time warping. In *Proc. 28th ACM SIGSPATIAL*, SIGSPATIAL '20, pages 99–110. ACM, 2020. doi:10.1145/3397536.3422245.

7 Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Lukas Plätz, Lea Thiel, and Sampson Wong. L-Budget Clustering (SWAT 24). Software, swhId: swh:1:dir:c28806ab33cb4a564e1d492efd322b1dc32f7245, (visited on 27/05/2024). URL: https://github.com/NathenMat/SWAT24.

8 Kevin Buchin, Anne Driemel, Joachim Gudmundsson, Michael Horton, Irina Kostitsyna, Maarten Löffler, and Martijn Struijs. Approximating (k, ℓ)-center clustering for curves. In *Proc. 30th SODA*, pages 2922–2938, 2019.

9 Kevin Buchin, Anne Driemel, Natasja van de L'Isle, and André Nusser. klcluster: Center-based clustering of trajectories. In *Proc. 27th ACM SIGSPATIAL*, pages 496–499, 2019.

10 Maike Buchin, Anne Driemel, and Dennis Rohde. Approximating (k,l)-median clustering for polygonal curves. *ACM Trans. Algorithms*, 19(1), February 2023. doi:10.1145/3559764.

11 Wai-sum Chan and Francis Yuk Lun Chin. Approximation of polygonal curves with minimum number of line segments or minimum error. *Int. J. Comput. Geom. Appl.*, 6:59–77, 1996.

12 Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. In *Proc. 29th STOC*, pages 626–635, 1997.

13 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985. doi:10.1016/0304-3975(85)90224-5.

14 Leonidas J Guibas, John E Hershberger, Joseph SB Mitchell, and Jack Scott Snoeyink. Approximating polygons and subdivisions with minimum-link paths. *Int. J. Comput. Geom. Appl.*, 3(04):383–415, 1993.

15 Dorit S. Hochbaum and David B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *J. ACM*, 33(3):533–550, 1986. doi:10.1145/5925.5933.

16 Hiroshi Imai and Masao Iri. Polygonal approximations of a curve — formulations and algorithms. In *Computational Morphology*, volume 6 of *Machine Intelligence and Pattern Recognition*, pages 71–86. North-Holland, 1988. doi:10.1016/B978-0-444-70467-2.50011-4.

17 Richard Mann, Robin Freeman, Michael Osborne, Roman Garnett, Chris Armstrong, Jessica Meade, Dora Biro, Tim Guilford, and Stephen Roberts. Objectively identifying landmark use and predicting flight trajectories of the homing pigeon using gaussian processes. *Journal of The Royal Society Interface*, 8(55):210–219, 2011.

18 Richard Matthew McCutchen and Samir Khuller. Streaming algorithms for k-center clustering with outliers and with anonymity. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 165–178, 2008. doi:10.1007/978-3-540-85363-3_14.

19 Majid Mirzanezhad. On approximate near-neighbors search under the (continuous) Fréchet distance in higher dimensions. *Information Processing Letters*, 183:106405, 2024. doi:10.1016/j.ipl.2023.106405.

20 Mees van de Kerkhof, Irina Kostitsyna, Maarten Löffler, Majid Mirzanezhad, and Carola Wenk. Global curve simplification. In *Proc. 27th ESA*, volume 144 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 67:1–67:14, 2019. doi:10.4230/LIPIcs.ESA.2019.67.

21 Marc van Kreveld, Maarten Löffler, and Lionov Wiratma. On optimal polyline simplification using the hausdorff and fréchet distance. *Journal of Computational Geometry*, 11(1):1–25, 2020. doi:10.20382/jocg.v11i1a1.