Temporal Clustering*†

Tamal K. Dey[1], Alfred Rossi[1], and Anastasios Sidiropoulos[1]

1 **Department of Computer Science and Engineering**
The Ohio State University. Columbus, OH, 43210.
{dey.8,sidiropoulos.1,rossi.49}@osu.edu

Abstract

We study the problem of clustering sequences of unlabeled point sets taken from a common metric space. Such scenarios arise naturally in applications where a system or process is observed in distinct time intervals, such as biological surveys and contagious disease surveillance. In this more general setting existing algorithms for classical (i.e. static) clustering problems are not applicable anymore.

We propose a set of optimization problems which we collectively refer to as *temporal clustering*. The quality of a solution to a temporal clustering instance can be quantified using three parameters: the number of clusters k, the spatial clustering cost r, and the maximum cluster displacement δ between consecutive time steps. We consider spatial clustering costs which generalize the well-studied k-center, discrete k-median, and discrete k-means objectives of classical clustering problems. We develop new algorithms that achieve trade-offs between the three objectives k, r, and δ. Our upper bounds are complemented by inapproximability results.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms, I.5.3 Clustering

Keywords and phrases clustering, multi-objective optimization, dynamic metric spaces, moving point sets, approximation algorithms, hardness of approximation.

Digital Object Identifier 10.4230/LIPIcs.ESA.2017.35

1 Introduction

Clustering points in a metric space is a fundamental problem that can be used to express a plethora of tasks in machine learning, statistics, and engineering, and has been studied extensively both in theory and in practice [4, 8, 13, 19, 20, 21, 23, 24, 26, 27, 29, 31]. Typically, the input consists of a set P of points in some metric space and the goal is to compute a partition of P minimizing a certain objective, such as the number of clusters given a constraint on their diameters.

We study the problem of clustering *sequences* of *unlabeled* point sets taken from a common metric space. Our goal is to cluster the points in each 'snapshot' so that the cluster assignments remain coherent across successive snapshots (across time). We formulate the problem in terms of tracking the *centers* of the clusters that may merge and split over time while satisfying certain constraints. Such instances are common in the study of time-evolving processes and phenomena under discrete observation. As an example consider a hypothetical study which aims to track the spread of a certain genetic mutation in plants. Here, data collection efforts center on annual field surveys in which a technician collects and catalogs samples. The location and number of mutation positive specimens change from year to

* This work was partially supported by the NSF grants CCF 1318595, CCF 1423230, DMS 1547357, and NSF award CAREER 1453472.
† See [9] for the full version of this paper.

year. Clustering such spaces is clearly a generalization of classical (static) clustering, which we refer to as *temporal clustering*. In this dynamic variant of the problem, apart from the number of clusters and their radii, we also wish to minimize the extent by which each cluster moves between consecutive snapshots.

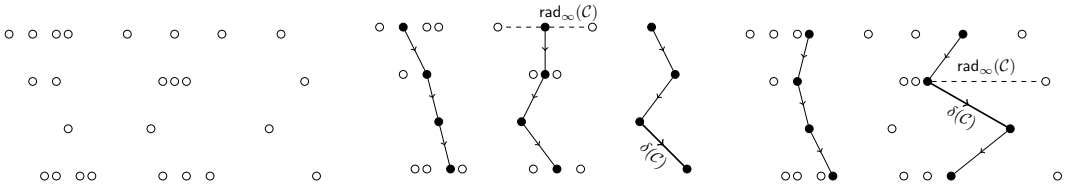
Related work. Clustering of moving point sets has been studied in the context of *kinetic clustering* [2, 6, 17, 18, 16, 14, 1]. In that setting points have identities (labels) which are fixed throughout their motion, the trajectories of the points are known beforehand, and the goal is to design a data structure which can efficiently compute a near-optimal clustering for any given time step. In our setting, since the points are not labeled there is, *a priori*, no explicit motion. Instead we are given a sequence of unlabeled points in a metric space and are required to assign the points of each to a limited number of temporally coherent clusters. Motion emerges as a consequence of cluster assignment. Consequently, kinetic clustering algorithms cannot be used in our setting. Another related problem concerns clustering time series under the Fréchet distance [11], with the clusters being constrained to move along polygonal trajectories of bounded complexity. This constraint is used to avoid overfitting, and is conceptually similar to our requirement that the clusters remain close between snapshots.

1.1 Problem formulations

Let us now formally define the algorithmic problems that we study in this paper. Perhaps surprisingly, very little is known for temporal clustering problems. There are of course different optimization problems that one could define; here we propose what we believe are the most natural ones.

We first define how the input to a temporal clustering problem is described. Let $M = (X, d)$ be a metric space. Let $P(1), \ldots, P(t)$ be a sequence of t finite, non-empty metric subspaces (points) of M. We refer to individual elements of this sequence (the 'snapshots') as *levels*, and collectively to P as a *temporal-sampling of* M of *length* t. The *size* of P is the total number of points over all levels, that is $\sum_{i\in[t]} |P(i)|$. Let $\{\tau(i)\}_{i=1}^{t}$ be a sequence of points such that $\tau(i) \in P(i)$ is a single point. We say that τ is a *trajectory* of P, and we let $\mathcal{T}(P)$ denote the set of all possible trajectories of P. For some $\mathcal{C} \subseteq \mathcal{T}(P)$, we denote by $\mathcal{C}(i)$ the set of points of the trajectories in $\mathcal{C}$ which lie in $P(i)$. In other words, $\mathcal{C}(i) = \bigcup_{\tau\in\mathcal{C}} \tau(i)$. The set of trajectories $\mathcal{C}$ induces a clustering on each level $P(i)$ by assigning each $p \in P(i)$ to the trajectory $\tau \in \mathcal{C}$ that minimizes $d(p, \tau(i))$. We refer to the points of $\mathcal{C}(i)$ as the *centers* of level i. Intuitively, this formulation allows points in different levels of P which are assigned to the same trajectory to be part of the same cluster; see Figure 1. Further, observe that trajectories may overlap allowing clusters to merge and split implicitly; see Figure 3a. We refer to $\mathcal{C}$ as a *temporal-clustering* of P.

We now formalize the clustering objectives. Our approach is to treat temporal clustering as a multi-objective optimization problem where we try to find a collection of trajectories such that their induced clustering ensures three conditions: (i) points in the same cluster remain near between successive levels (*locality*), (ii) the restriction of the clustering to any single level fits the shape of the data (*spatial constraint*), and (iii) we do not return excessively many clusters (*complexity*). To measure how far some trajectory τ jumps, we define its *displacement*, denoted by $\delta(\tau)$, to be $\delta(\tau) = \max_{i\in[t-1]} d(\tau(i), \tau(i+1))$. We also define the displacement of $\mathcal{C}$ to be $\delta(\mathcal{C}) = \max_{\tau\in\mathcal{C}} \delta(\tau)$. Finally, we consider three different objectives for the spatial cost, which correspond to generalization of the k-center, k-median, and k-means respectively. The first one, corresponding to k-center, is the maximum over all levels of the maximum cluster radius; formally $\mathsf{rad}_\infty(\mathcal{C}) = \max_{i\in[t]} \max_{p\in P(i)} d(p, \mathcal{C}(i))$,



(a) P **(b)** A temporal 3-clustering **(c)** A temporal 2-clustering

Figure 1 (1a) A temporal-sampling P of length 4 where $P(i) \subset \mathbb{R}$ is drawn horizontally. Each level of P is depicted as a row starting from $P(1)$ at the top. (1b) The temporal-sampling P shown with a clustering $\mathcal{C}$, consisting of 3 clusters. The centers of each of 3 trajectories are depicted as filled circles in each level. Arrows are drawn between $\tau_j(i)$ and $\tau_j(i+1)$ for each trajectory τ_j, $j \in \{1,2,3\}$. In the first level, a pair of points which achieve the spatial cost are joined to their respective cluster centers by a dashed edge. The arrow between the pair of centers which achieves maximum displacement is shown in bold. (1c) The temporal-sampling P shown with 2 clusters.

where $d(p,\mathcal{C}(i)) = \min_{\tau\in\mathcal{C}} d(p,\tau(i))$. The second and third spatial cost objectives, which corresponding to discrete k-median, and discrete k-means (respectively), are defined to be $\mathsf{rad}_1(\mathcal{C}) = \max_{i\in[t]} \sum_{p\in P(i)} d(p,\mathcal{C}(i))$, and $\mathsf{rad}_2(\mathcal{C}) = \max_{i\in[t]} \sum_{p\in P(i)} d(p,\mathcal{C}(i))^2$.

▶ **Definition 1.** Let $r \in \mathbb{R}_{\geq 0}$, $\delta \in \mathbb{R}_{\geq 0}$. We say that a set of trajectories $\mathcal{C} \subseteq \mathcal{T}(P)$ is a *temporal* (k,r,δ)*-clustering* of P if $\mathsf{rad}_\infty(\mathcal{C}) \leq r$, $\delta(\mathcal{C}) \leq \delta$, and $|\mathcal{C}| \leq k$. (See Figure 1 for an example.) We further define *temporal* (k,r,δ)*-median-clustering* and (k,r,δ)*-means-clustering* analogously by replacing rad_∞ by rad_1 and rad_2 respectively.

We now formally define the optimization problems that we study. In the case of static clustering, a natural objective is to minimize the maximum cluster radius, subject to the constraint that only k clusters are used; this is the classical k-Center problem [23]. Another natural objective in the static case is to minimize the number of clusters subject to the constraint that the radius of each cluster is at most r, for some given $r > 0$; this is the r-Dominating Set problem [22]. Our definition of temporal clustering includes the temporal analogues of k-Center and r-Dominating Set as special cases.

▶ **Definition 2** (Temporal (k,r,δ)-Clustering problem)**.** An instance of the Temporal (k,r,δ)-Clustering problem is a tuple (M,P,k,r,δ), where M is a metric space, P is a temporal-sampling of M, $k \in \mathbb{N}$, $r \in \mathbb{R}_{\geq 0}$, and $\delta \in \mathbb{R}_{\geq 0}$. The goal is to decide whether P admits a temporal (k,r,δ)-clustering.

▶ **Definition 3** (Temporal (k,r,δ)-Clustering approximation)**.** Given an instance of the Temporal (k,r,δ)-Clustering problem consisting of a tuple (M,P,k,r,δ), a (α,β,γ)-*approximation* is an algorithm which either returns a temporal $(\alpha k,\beta r,\gamma\delta)$-clustering of P, or correctly decides that no temporal (k,r,δ)-clustering exists. In general α, β, and γ can be functions of the input.

We analogously define the Temporal (k,r,δ)-Median Clustering problem and approximation, and the Temporal (k,r,δ)-Means Clustering problem and approximation by replacing in Definitions 2 and 3 $(\cdot,\cdot,\cdot)$-clustering by $(\cdot,\cdot,\cdot)$-median-clustering and $(\cdot,\cdot,\cdot)$-means-clustering respectively.

1.2 Our contribution

To the best of our knowledge, this is the first study of the above models of temporal clustering. Our main contributions consist of polynomial-time approximation algorithms for several

temporal clustering variants, and hardness of approximation results for others.

Temporal clustering. We begin by discussing our results on Temporal (k,r,δ)-Clustering. We first consider the problem of minimizing r and δ while keeping k fixed. This is a generalization of the static k-Center problem. We present a polynomial-time $(1, 2, 1 + 2\varepsilon)$-approximation algorithm where $\varepsilon = r/\delta$ using a different method. More specifically, our result is obtained via a reduction to a network flow problem. We show that the problem is NP-hard to approximate to within polynomial factors even if we increase the radius by a polynomial factor. Formally, we show that it is NP-hard to obtain a $(1, \mathsf{poly}(n), \mathsf{poly}(n))$-approximation.

Next we consider the problem of minimizing the number of clusters k, while fixing r and δ. This is a generalization of the static r-Dominating Set problem. We obtain a polynomial-time $(\ln n, 1, 1)$-approximation algorithm. For the static case, the polynomial-time $(\ln n)$-approximation algorithm follows by a reduction to the Set-Cover problem, and is known to be best-possible [10, 30, 12]. However, in the temporal case, this reduction produces an instance of Set-Cover of exponential size. Thus, it does not directly imply a polynomial-time algorithm for Temporal r-Dominating Set. We bypass this obstacle by showing how to run the greedy algorithm for Set-Cover on this exponentially large instance in polynomial-time, without explicitly computing the Set-Cover instance. We also argue that $(\ln n, 1, 1)$-approximation is best possible by observing that $((1-\varepsilon)\ln n, 2-\varepsilon', \cdot)$-approximation is NP-hard for any $\varepsilon, \varepsilon' > 0$.

We further present a result that can be thought of as a trade-off between the above two settings by allowing both the number of clusters and the radius to increase. More precisely, we obtain a polynomial-time $(2, 2, 1 + \varepsilon)$-approximation algorithm where $\varepsilon = r/\delta$. Interestingly, we can show that obtaining a $(1.005, 2 - \varepsilon, \mathsf{poly}(n))$-approximation is NP-hard.

The following summarizes the above approximation algorithms.

▶ **Theorem 4.** Temporal (k,r,δ)-Clustering *admits the following algorithms:*
(1) $(1,2,1+2\varepsilon)$*-approximation where* $\varepsilon = r/\delta$*,*
(2) $(\ln(n),1,1)$*-approximation,*
(3) $(2,2,1+\varepsilon)$*-approximation where* $\varepsilon = r/\delta$*,*
where n *is the size of the temporal-sampling. Moreover, the running time of all of these algorithms is* $O(n^3)$*.*

We prove Theorems 4.1, 4.2, 4.3 in Sections 2.1, 2.2, 2.3, respectively.

It is important that the approximation in displacements for Theorem 4.1 and Theorem 4.3 takes into account the factor $\varepsilon = r/\delta$ if a polynomial time algorithm is aimed for. This is because our inapproximability results as summarized below show that the problem is NP-hard otherwise.

▶ **Theorem 5.** *The status of* Temporal (k,r,δ)-Clustering *with temporal-samplings of size* n *is as follows:*
(1) There exist universal constants $c > 0$*,* $c' > 0$ *such that* $(1,cn^{s(1-\varepsilon)},c'n^{(1-s)(1-\varepsilon)})$*-approximation is* NP*-hard for any* $\varepsilon, s \in \mathbb{R}$ *where* $\varepsilon > 0$ *and* $s \in [0,1]$*.*
(2) $((1-\varepsilon)\ln(n),2-\varepsilon',\cdot)$*-approximation is* NP*-hard for any fixed* $\varepsilon > 0$*,* $\varepsilon' > 0$*.*
(3) There exists a universal constant c *such that* $(1.00579,2-\varepsilon',cn^{1-\varepsilon})$*-approximation is* NP*-hard for any fixed* $\varepsilon > 0$*,* $\varepsilon' > 0$*.*
Moreover, items 5.1 and 5.3 remain NP*-hard even for temporal-samplings in* 2*-dimensional Euclidean space.*

Due to space constraints, we defer extended discussion of Theorem 5.1, Theorem 5.2, and Theorem 5.3 to the full version of the paper [9].

Temporal median clustering. We next discuss our result on the Temporal (k,r,δ)-Median Clustering problem. The static k-Median problem admits an $O(1)$-approximation via local search [5, 28]. In Section 2.4 we show that the local search approach fails in the temporal case, even on temporal samplings of length two. We present an algorithm that achieves a trade-off between the number of clusters and the spatial cost. The result is obtained via a greedy algorithm, which is similar to the one used for the k-Set Cover problem. The result is summarized in the following theorem.

▶ **Theorem 6.** *For any fixed $\varepsilon > 0$, there exists an $(O(\log(n\Delta/\varepsilon)),\ 1+\varepsilon, 1)$-median-approximation algorithm with running time $\mathsf{poly}(n, \log(\Delta/\varepsilon))$, on an instance of size n and a metric space of spread Δ.*

The result is obtained by iteratively selecting a trajectory which minimizes a certain potential function. The proof uses submodularity and monotonicity of the potential function. These properties remain true if the potential function is modified by replacing $d(p,\mathcal{C}(i))$ with $d(p,\mathcal{C}(i))^2$, and thus an identical theorem holds for Temporal k-Means.

We complement the above algorithm by showing the following hardness result.

▶ **Theorem 7.** *The status of* Temporal (k,r,δ)-Median Clustering *with temporal-samplings of size n is as follows:*

(1) There exist universal constants c_r, c_δ such that $(1, c_r n^{s(1-\varepsilon)}, c_\delta n^{(1-s)(1-\varepsilon)})$-approximation for Temporal k-Median *is* NP*-hard for any $\varepsilon, s \in \mathbb{R}$ where $\varepsilon > 0$ and $s \in [0,1]$.*

(2) Let c, s be the constants from Theorem 4.6 (3) in [7]. Let $0 \le f < c - s$. Then $(\frac{3-(s+f)}{3-c}, 1+c_r f, c_\delta n^{1-\varepsilon})$-approximation is NP*-hard for any fixed $\varepsilon > 0$ and some constants c_r, c_δ.*

Moreover, item 7.1 remains hard even for temporal-samplings from 2-dimensional Euclidean space.

The clustering instances used in the proofs of Theorem 7.1 and Theorem 7.2 involve clusterings which use only a constant number of points per cluster, thus the same constructions suffice to prove hardness of Temporal (k,r,δ)-Means Clustering with only slight modification of the distances. See the discussion in the full version [9].

Additional notation and preliminaries. Let $r > 0$. An r-net in some metric space (X,d) is some maximal $Y \subseteq X$, such that for any $x, y \in Y$, with $x \neq y$, we have $d(x,y) > r$. Let P be a temporal-sampling of length t in some metric space (X,d). Let $V(P,i) = \bigcup_{x \in P(i)} \{(i,x)\}$ for all $i \in [t]$. For any trajectory τ, and for any $r \ge 0$, the *tube* around τ of *radius* r, denoted by $\mathsf{tube}(\tau, r)$, is defined to be $\mathsf{tube}(\tau, r) = \bigcup_{i \in [t]} \{(i,x) \in V(P,i) \mid x \in \mathsf{ball}(\tau(i), r)\}$, where for $x \in X$, $r \in \mathbb{R}_{\ge}0$, we use the notation $\mathsf{ball}(x,r)$ to denote a closed ball of radius r. Let $\delta \in \mathbb{R}_{\ge 0}$. The directed graph $G_\delta(P)$ has as vertices $V(P,i)$ for all $i \in [t]$. For any $i \in [t-1]$ there is an edge between $p \in V(P,i)$ and $q \in V(P,i+1)$ whenever $d(p,q) \le \delta$ (see Figure 2).

2 Algorithms

2.1 Exact number of clusters: $(1,2,1+2\varepsilon)$-approximation

In this section, we consider the problem of computing a temporal clustering by relaxing the radius and the displacement, while keeping the number of clusters exact. This is a temporal analogue of the k-Center problem. We first present a polynomial time $(1,2,1+2\varepsilon)$-approximation where $\varepsilon = r/\delta$. In the full version [9], we complement this with an inapproximability result.

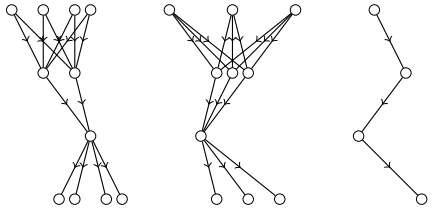


Figure 2 The graph $G_\delta(P)$ for P from the previous diagram and some δ. Points which are within a distance of δ in adjacent levels are connected by a directed edge which points toward the higher indexed level.

An auxiliary network flow problem. The high-level idea of the polynomial time algorithm is to use a reduction to a specific network flow problem. Specifically, we seek a minimum flow which satisfies lower bound constraints along certain edges. This is the so-called minimum flow, or minimum feasible flow problem [3, 15]. We now formally define this flow network. For each $i \in [t]$, let $C(i) \subseteq P(i)$. Let $\gamma > 0$. We construct a flow network, denoted by $N_\gamma(P, C)$ where C is the sequence of centers $C(i)$ for $i \in [t]$. We start with the graph $G_\gamma(P)$. In level i, we replace each vertex $v = (i, c)$ for $c \in C(i)$ by a pair of vertices $\mathsf{tail}(v)$ and $\mathsf{head}(v)$, and we connect them by an edge $(\mathsf{tail}(v), \mathsf{head}(v))$. For vertices $v = (i, p)$ where $p \in P(i) \setminus C(i)$ we define $\mathsf{tail}(v) = \mathsf{head}(v) = v$. Now for *any* vertex v, all incoming edges to v become incoming edges to $\mathsf{tail}(v)$, and all outgoing edges from v become outgoing edges from $\mathsf{head}(v)$. We add a source vertex s and a sink vertex s'. For all $p \in P(1)$, we add an edge from s to $\mathsf{tail}((1, p))$. Similarly, for all $p \in P(t)$, we add an edges from $\mathsf{head}((t, p))$ to s'. We set the capacity of each edge to be ∞. Finally, we set a lower bound of 1 to the capacity of every edge $(\mathsf{tail}(v), \mathsf{head}(v))$, for all $v = (i, c)$, $c \in C(i)$, $i \in [t]$ (see Figure 3b).

Algorithm. We first compute a net at every level of the temporal-sampling and then we reduce the problem of computing a temporal clustering to a flow instance, using the network flow defined above. By computing an integral flow and decomposing it into paths, we obtain a collection of trajectories. The lower bound constraints ensure that all net points are covered; this allows us to show that all points are covered by the tubular neighborhoods of the trajectories. Formally, the algorithm consists of the following steps:

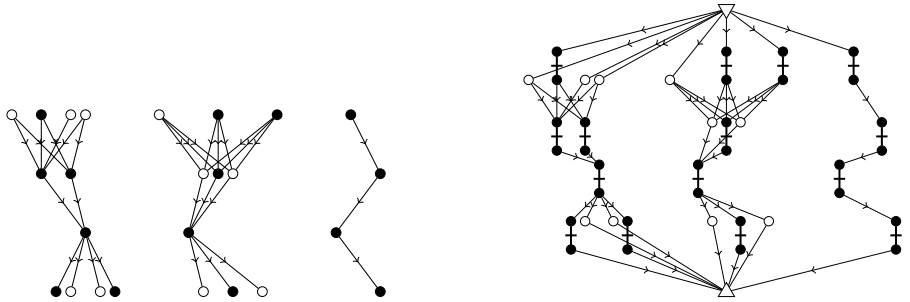
Step 1: Computing nets. For each $i \in [t]$, compute a $2r$-net $C(i)$ of $P(i)$. If for some $i \in [t]$, $|C(i)| > k$, then return nil.

Step 2: Constructing a flow instance. We construct the minimum flow instance $N_{2r+\delta}(P, C)$.

Step 3: Computing a collection of trajectories. If the flow instance $N_{2r+\delta}(P, C)$ is not feasible, then return nil. Otherwise, find a minimum integral flow F in $N_{2r+\delta}(P, C)$, satisfying all the lower bound constraints. Decompose F into a collection of paths, each carrying a unit of flow. The restriction of each path in G is a trajectory. Output the set of all these trajectories.

Throughout the rest of this section let P be a temporal-sampling. We now show that if there exists a temporal (k,r,δ)-clustering, then the above algorithm outputs a temporal $(k, 2r, (1 + 2\varepsilon)\delta)$-clustering where $\varepsilon = r/\delta$.

▶ **Lemma 8.** *Suppose that P admits a temporal (k,r,δ)-clustering, $\mathcal{Q}$. For each $i \in [t]$ let $Q(i)$ denote the level i centers of $\mathcal{Q}$, and let $C(i)$ be a $2r$-net of $P(i)$. Then the map $\pi_i : C(i) \to Q(i)$ which sends each $2r$-net center to a nearest center in $Q(i)$ is injective.*



(a) The graph $G_\gamma(P)$. **(b)** The flow network $N_\gamma(P, C)$.

Figure 3 (3a) The graph $G_\gamma(P)$ for $P(i) \subset \mathbb{R}$ and some $\gamma > 0$. The vertices in $C(i)$, $i \in [4]$, are indicated with filled circles. (3b) The flow network $N_\gamma(P, C)$ corresponding to $G_\gamma(P)$. Every node from C has been split into an edge.

Proof. First, observe that for each $c \in C(i)$, $d(c, \pi_i(c)) \leq r$ because r-balls centered at the points in $Q(i)$ cover $P(i)$ and hence $C(i)$. For injectivity of π_i, observe that, $\pi_i(c) \neq \pi_i(c')$ for $c \neq c'$ because otherwise the inequality $d(c, c') \leq d(c, \pi_i(c)) + d(c', \pi_i(c')) \leq 2r$ holds violating the property that $C(i)$ is a $2r$-net. ◀

Since for each $i \in [t]$, the map π_i is injective, it follows that $|C(i)| \leq |Q(i)| \leq k$. So, we have the following immediate Corollary.

▶ **Corollary 9.** *If P admits a temporal (k,r,δ)-clustering then for any $i \in [t]$, any $2r$-net $C(i)$ of $P(i)$ has $|C(i)| \leq k$.*

▶ **Lemma 10.** *If P admits a temporal (k,r,δ)-clustering then for any level-wise $2r$-net C, the flow instance $N_{2r+\delta}(P, C)$ admits a feasible flow of value k.*

Proof. Fix a temporal (k,r,δ)-clustering $\mathcal{Q}$ and let τ denote one of its k trajectories. The graph $G_{2r+\delta}(P)$ contains a path corresponding to τ as the distance between any pair of consecutive points in P is at most δ. For each i, let $\pi_i : C(i) \to Q(i)$ denote a map which sends each $2r$-center of $C(i)$ to a nearest center in $Q(i)$. We modify τ to produce some path τ' in $G_{2r+\delta}(P)$ as follows: for every level i such that $\tau(i) = \pi_i(c_i)$ for some net-point $c_i \in C(i)$ we let $\tau'(i) = c_i$, otherwise we set $\tau'(i) = \tau(i)$. We observe that in the worst case the distance between consecutive points, say $u = \tau'(i)$ and $v = \tau'(i+1)$, is at most $2r + \delta$ because of the following inequality (see Figure 4) $d(u, v) \leq d(u, \tau(i)) + d(\tau(i), \tau(i+1)) + d(\tau(i+1), v) \leq r + \delta + r$. It follows that τ' is indeed a path in $G_{2r+\delta}(P)$. Further, by the injectivity of each map π_i (Lemma 8) which is used in deforming τ to τ', we have that for every net point, there exists some τ' that contains it. In other words, all net points $C(i)$ are covered by the paths τ'. For each optimal trajectory τ, let τ'' be the path in $N_{2r+\delta}(P, C)$ obtained from τ' by connecting s to the first vertex in τ', and the last vertex in τ' to t. By routing a unit of flow in $N_{2r+\delta}(P, C)$ along each such τ'' we obtain a flow of value at most k that meets all the demands along the edges corresponding to net points C, concluding the proof. ◀

▶ **Lemma 11.** *Given k, r, δ, and a temporal-sampling P, with $|P| = n$, there exists an $O(n^3)$-time algorithm that either correctly decides P does not admit a temporal (k,r,δ)-clustering, or outputs some temporal $(k,2r,2r+\delta)$-clustering.*

Proof. Lemmas 9 and 10 imply that if a temporal (k,r,δ)-clustering exists, then the algorithm does not return nil, and thus outputs a set T of at most k trajectories. Let $\mathcal{C}$ be the temporal

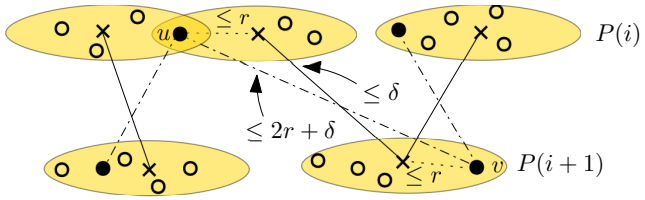


Figure 4 The crosses, filled circles, and empty circles are optimal centers, net points, and other points respectively. The paths (τ) in optimal solution and the deformed paths(τ') are indicated with solid and dotted edges respectively.

clustering corresponding to T. Each trajectory in T corresponds to a path in $G_{2r+\delta}(P)$, thus has displacement at most $2r+\delta$. Therefore $\delta(\mathcal{C}) \leq 2r+\delta$. Since F is a feasible flow, it follows that all lower bound constraints in $N_{2r+\delta}(P, C)$ are satisfied. Thus for all $i \in [t]$, for all $c \in C(i)$, there exists at least one unit of flow along the edge $(\mathsf{tail}(v), \mathsf{head}(v))$ corresponding to the vertex $v = (i, c)$; it follows that there exists some trajectory containing c in level i. Since for all $i \in [t]$, $C(i)$ is a $2r$-net of $P(i)$, it follows that $P(i) \subseteq \bigcup_{c \in C(i)} \mathsf{ball}(c, 2r)$. Thus $\bigcup_{i \in [t]} V(P, i) \subseteq \bigcup_{\tau \in T} \mathsf{tube}(\tau, 2r)$, which implies that $\mathsf{rad}_\infty(\mathcal{C}) \leq 2r$. We thus obtain that $\mathcal{C}$ is a temporal $(k,2r,2r+\delta)$-clustering. Finally, we bound the running time. Computing the $2r$-nets over all levels, checking their sizes can be done in $O(nk)$ time. Building $G_{2r+\delta}(P)$ and $N_{2r+\delta}(P, C)$ can be done in $O(n^2)$ time. Finding an integral solution to $N_{2r+\delta}(P, C)$ takes $O(n^3)$ time using the algorithm of Gabow and Tarjan [15]. Decomposing the resulting flow takes $O(n^3)$ time. We conclude that the entire procedure completes in $O(n^3)$ time. ◀

Writing $\varepsilon = r/\delta$, we immediately obtain Theorem 4.1 from Lemma 11.

2.2 Exact radius and displacement: $(\ln(n),1,1)$-approximation

In this section we consider the case where the number of clusters is allowed to be approximated in analogy to the static r-Dominating Set problem. We present a polynomial-time $(\ln(n),1,1)$-approximation algorithm. In the full version [9], we argue that this result is tight in the sense that obtaining a $((1-\varepsilon)\ln(n),1,1)$-approximation is NP-hard for any fixed $\varepsilon > 0$.

Let P be a temporal-sampling of length t. For any $\delta \geq 0$, we denote by $\mathcal{T}_\delta(P)$ the set of all trajectories of displacement at most δ. Given an instance of the Temporal (k,r,δ)-Clustering problem consisting of a tuple (M, P, k, r, δ), the high level idea is to express the problem as an instance of Set-Cover. Recall that an instance of Set-Cover consists of a pair $(U, \mathcal{S})$, where U is a set, and $\mathcal{S}$ is a collection of subsets of U. The goal is to find some $\mathcal{S}' \subseteq \mathcal{S}$, minimizing $|\mathcal{S}'|$, such that $U \subseteq \bigcup_{X \in \mathcal{S}'} X$, if such $\mathcal{S}'$ exists. We set $U = \bigcup_{i \in [t]} V(P, i)$, and $\mathcal{S} = \bigcup_{\tau \in \mathcal{T}_\delta(P)} \{\mathsf{tube}(\tau, r)\}$. We will show that a solution to the Set-Cover instance $(U, \mathcal{S})$ can be used to obtain a temporal $(\ln(n)k,r,\delta)$-clustering. Note that $\mathcal{S}$ can have cardinality exponential in the size of the input. However, as we shall see, we can still obtain an approximate solution for $(U, \mathcal{S})$ in polynomial-time.

We first establish that any $\alpha(n)$-approximate solution to $(U, \mathcal{S})$ can be converted, in polynomial-time, to a temporal $(\alpha(n)k,r,\delta)$-clustering. Let s_{OPT} denote the minimum cardinality of any feasible solution for $(U, \mathcal{S})$ when it exists. Similarly, let k_{OPT} denote the smallest value of k' such that P admits a temporal (k',r,δ)-clustering.

▶ **Lemma 12.** $k_{\mathrm{OPT}} = s_{\mathrm{OPT}}$.

The proof of Lemma 12 is deferred to the full version [9]. We next establish the following result which allows us to run the greedy algorithm for Set-Cover on the instance $(U, \mathcal{S})$ in

polynomial-time, even though $|\mathcal{S}|$ can be exponentially large.

▶ **Lemma 13.** *Let $\mathcal{S}' \subsetneq \mathcal{S}$. There exists an $O(n^2)$ time algorithm which computes some $X \in \mathcal{S} \setminus \mathcal{S}'$, maximizing $\left|X \cap \left(U \setminus \bigcup_{Y \in \mathcal{S}'} Y\right)\right|$. Moreover, the algorithm outputs some trajectory $\tau \in \mathcal{T}_\delta(P)$, such that $X = \{\mathsf{tube}(\tau, r)\}$.*

The proof of Lemma 13 is deferred to the full version [9]. We are now ready to prove Theorem 4.2.

Proof of Theorem 4.2. Recall that the classical greedy algorithm for Set-Cover computes a solution $\mathcal{S}' \subseteq \mathcal{S}$, if one exists, as follows: Initially, we set $\mathcal{S}' = \varnothing$. At every iteration, we pick some $X \in \mathcal{S} \setminus \mathcal{S}'$ such that $\left|X \cap \left(U \setminus \bigcup_{Y \in \mathcal{S}'} Y\right)\right|$ is maximized, and we add X to $\mathcal{S}$. The algorithm stops when either U is covered by $\mathcal{S}$, or when no further progress can be made, i.e. when $\left|X \cap \left(U \setminus \bigcup_{Y \in \mathcal{S}'} Y\right)\right| = 0$; in the latter case, the instance $(U, \mathcal{S})$ is infeasible. It is well-known that this algorithm achieves an approximation ratio of $\ln n$ for Set-Cover [25]. Now if $(U, \mathcal{S})$ is infeasible the above procedure detects this and terminates. Otherwise, let $\mathcal{S}' \subseteq \mathcal{S}$ be the feasible solution found by repeatedly using the procedure described in Lemma 13. The corresponding trajectories returned by this procedure form a temporal (k',r,δ)-clustering of P, for some $k' = |\mathcal{S}'| \leq \ln n \cdot s_{\mathrm{OPT}}$. By Lemma 12 it follows that $k' \leq \ln n \cdot k_{\mathrm{OPT}} \leq \ln n \cdot k$. Thus we obtain an $(\ln(n),1,1)$-approximation. Finally, to bound the running time note that in the worst case, the total number of calls to the procedure in Lemma 13 is n since at every step we cover at least uncovered point. The theorem now follows by the fact that each call takes $O(n^2)$ time. ◀

2.3 Approximating all parameters: $(2,2,1+\varepsilon)$-approximation

So far we have constrained either the number of clusters or the radius and the displacement to be exact. We now describe an algorithm that relaxes all three parameters simultaneously. We present a polynomial-time $(2, 2, 1 + \varepsilon)$-approximation algorithm where $\varepsilon = r/\delta$. We complement this solution in the full version [9] by showing that it is NP-hard to obtain a $(1.005, 2 - \varepsilon, \mathsf{poly}(n))$-approximation for any $\varepsilon > 0$.

▶ **Lemma 14.** *If P admits a temporal (k,r,δ)-clustering then for any level-wise $2r$-net C, the flow instance $N_{r+\delta}(P, C)$ admits a feasible flow of value $2k$.*

Proof. Fix a temporal (k,r,δ)-clustering $\mathcal{C} = \{\tau_i\}_{i=1}^k$. We inductively define a sequence $\mathcal{Q}_0, \ldots, \mathcal{Q}_t$, where for each $i \in \{0, \ldots, t\}$, $\mathcal{Q}_i$ is a multiset of paths in $G_{r+\delta}(P)$. We set $\mathcal{Q}_0 = \{\sigma_1^1, \sigma_1^2, \ldots, \sigma_k^1, \sigma_k^2\}$, where for each $j \in [k]$, we have $\sigma_j^1 = \sigma_j^2 = \tau_j$. Next, we inductively define $\mathcal{Q}_i$, for some $i \in \{1, \ldots, t\}$. Starting with $\mathcal{Q}_i = \mathcal{Q}_{i-1}$, we proceed to modify $\mathcal{Q}_i$. By induction, it follows that the paths σ_j^1, σ_j^2, and τ_j share the same suffix at levels $i, \ldots, t$. Thus, $\tau_j(i) \in \sigma_j^1$ and $\tau_j(i) \in \sigma_j^2$. Now, for the modification, we consider each $c \in C(i)$, and proceed as follows (see Figure 5 for an illustration). Since $\mathcal{C}$ is a valid temporal (k,r,δ)-clustering, it follows from Lemma 8 that there exists an injective map π_i from $C(i)$ to the set $\tau_1(i), \ldots, \tau_k(i)$ so that $\pi_i(c) = \tau_j(i)$ for some $j \in [k]$ and $d(\tau_j(i), c) \leq r$. We consider the following two cases:

Case 1: If i is odd and $\tau_j(i) = \pi_i(c)$ for some $c \in C(i)$, then we modify σ_j^1 by replacing the vertex $\tau_j(i)$ with c.

Case 2: If i is even and $\tau_j(i) = \pi_i(c)$ for some $c \in C(i)$, then we modify σ_j^2 by replacing the vertex $\tau_j(i)$ with c.

We next argue that the result is indeed a path in $G_{r+\delta}(P)$. Suppose that in the above step, we modify the path σ_j^ℓ, for some $\ell \in \{1, 2\}$ so that $\sigma_j^\ell(i) = c$. It follows by induction on i that the path σ_j^ℓ was not modified when constructing $\mathcal{Q}_{i-1}$; thus $\sigma_j^\ell(i-1) = \tau_j(i-1)$. Since

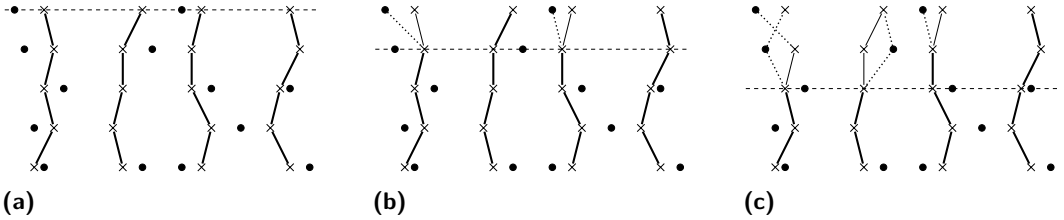


Figure 5 An example of the inductive construction of the multisets of paths $\mathcal{Q}_i$, for $i = 0$ (Figure 5a), $i = 1$ (Figure 5b), and $i = 2$ (Figure 5c). Dotted lines show where a trajectory has been rounded to a net point. Thin and thick solid lines indicate where one or two trajectories are coincident to an optimal trajectory, respectively. Initially (Figure 5a), $\mathcal{Q}_0$ consists of $2k$ trajectories $\sigma_j^1 = \sigma_j^2 = \tau_j$, for the trajectories of some optimal solution $\tau_1, \ldots, \tau_k$. At step i, for any $j \in [k]$ such that $\tau_j(i)$ is within a distance of r from some net point, c, we obtain $\mathcal{Q}_i$ by replacing $\tau_j(i)$ with c in either σ_j^1 or σ_j^2, depending on the parity of i.

$\delta(\tau_j) \leq r$, it follows by the triangle inequality that $d(\sigma_j^\ell(i-1), \sigma_j^\ell(i)) = d(\tau_j(i-1), c) \leq d(\tau_j(i-1), \tau_j(i)) + d(\tau_j(i), c) \leq \delta + r$. It follows that $\delta(\sigma_j^\ell) \leq r + \delta$, which implies that each element of $\mathcal{Q}_i$ is indeed a path in $G_{r+\delta}(P)$. This completes the inductive definition of the multisets $\mathcal{Q}_0, \ldots, \mathcal{Q}_t$. It is immediate by induction that for each $i \in [t]$, for each $c \in C(i)$, there exist some path $\sigma \in \mathcal{Q}_t$ that visits c. We next transform the collection $\mathcal{Q}_t$ into a flow F in $N_{r+\delta}(P, C)$. For each path $\sigma \in \mathcal{Q}_t$, we obtain a path in the network $N_{r+\delta}(P, C)$ starting from the source s, then replacing for each $i \in [t]$, each $c \in C(i) \cap \sigma$ by the edge $(\mathsf{tail}(v), \mathsf{head}(v))$, for $v = (i, c)$, then terminating at the sink s'; we route a unit of flow along the resulting path. Since for each $i \in [t]$, there exists some path in $\mathcal{Q}_t$ visiting each $c \in C(i)$, it follows that all lower-bound constraints in $N_{r+\delta}(P, C)$ are satisfied by F. Since $\mathcal{Q}_t$ contains $2k$ paths, it follows that the value of the resulting flow is $2k$, as required. ◀

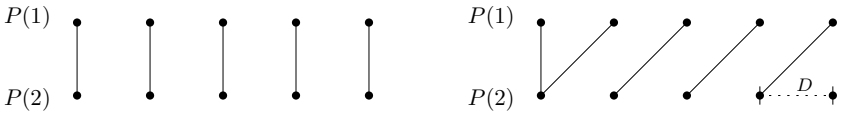
We are now ready to prove Theorem 4.3.

Proof of Theorem 4.3. For each $i \in [t]$, compute a $2r$-net $C(i)$ of $P(i)$, and construct the flow network $N_{r+\delta}(P, C)$. Compute a minimum flow F in $N_{r+\delta}(P, C)$ satisfying all lower-bound constraints. If $N_{r+\delta}(P, C)$ is infeasible (i.e. if there is no flow satisfying all lower bound constraints), or if the the value of the minimum flow in $N_{r+\delta}(P, C)$ is greater than $2k$, it follows by Lemma 14 that P does not admit a temporal (k,r,δ)-clustering. Thus, in this case the algorithm terminates. Otherwise, we compute a minimum flow in $N_{r+\delta}(P, C)$. Since all capacities and lower-bound constraints in $N_{r+\delta}(P, C)$ are integers, it follows that F can be taken to be integral. We decompose F into a collection of at most $2k$ paths, each carrying a unit of flow. Arguing as in Lemma 11 we have that the restriction of these paths on $G_{r+\delta}(P)$ is a set of trajectories that induces a valid temporal $(2k,2r,r+\delta)$-clustering of P. This provides a $(2,2,1+\varepsilon)$-approximation where $\varepsilon = r/\delta$. Finally, the running time is easily seen as $O(n^3)$ by the same argument that appears in Lemma 11, concluding the proof. ◀

2.4 Approximation algorithm for temporal median clustering

In this section we consider variants of Temporal Clustering which evaluate the spatial cost of clustering by taking the level-wise maximum of discrete k-median and discrete k-means objectives. A natural question is whether or not the problem admits a $O(1)$-approximation via local search, as in static case [5, 28]. In Figure 6 we show that the local search approach fails, even on temporal samplings of length two. Instead, the result is obtained by iteratively selecting a trajectory which most improves a certain potential function. The result in this

section is presented for the Temporal (k,r,δ)-Median Clustering problem, and follows by submodularity and monotonicity of the potential function. These properties remain if $d(p,\mathcal{C}(i))$ is replaced with the $d(p,\mathcal{C}(i))^2$, and thus holds identically for Temporal k-Means.



(a) The clustering $\mathcal{C}^*$. **(b)** The clustering $\mathcal{C}$.

Figure 6 An example demonstrating that local search fails. Consider a temporal-sampling P of length 2 where $P(1) = P(2)$ consists of a sequence of 5 points where successive points are separated by D. (6a) A temporal $(5, r, \delta)$-median-clustering, $\mathcal{C}^*$, for any $r, \delta \in \mathbb{R}_{\geq 0}$ with $\mathsf{rad}_1(\mathcal{C}^*) = 0$. (6b) A temporal $(5, D, \delta)$-median-clustering, $\mathcal{C}$, for any $D \leq \delta < 2D$. Note that swapping any trajectory in $\mathcal{C}$ with one in $\mathcal{T}_\delta(P)$ is non-improving. The clustering $\mathcal{C}$ is therefore a local minimum of local search, yet the ratio $\mathsf{rad}_1(\mathcal{C})/\mathsf{rad}_1(\mathcal{C}^*)$ remains unbounded.

We now present an approximation algorithm for the Temporal (k,r,δ)-Median Clustering problem. Let $\mathcal{I} = (M, P, k, r, \delta)$ be an input to the problem, where P is a temporal-sampling of length t. Let n denote the size of the P. Let also Δ denote the spread of $M = (X, d)$. That is, $\Delta = \frac{\operatorname{diam}(M)}{\inf_{p,q \in X}\{d(p,q):d(p,q)>0\}}$. Since we only consider finite metric spaces, and since the single point case is trivial, w.l.o.g. we may assume that the diameter of M is Δ and minimum interpoint distance in M is 1. For a set of trajectories $\mathcal{C}$ we define $\mathsf{cost}(i;\mathcal{C}) = \sum_{p\in P(i)} d(p,\mathcal{C}(i))$. We also define $W(\mathcal{C}) = \sum_{i=1}^{t} \max\{0, \mathsf{cost}(i;\mathcal{C}) - r\}$. Intuitively, the quantity $W(\mathcal{C})$ measures how far the solution $\mathcal{C}$ is from the optimum; in particular, if $W(\mathcal{C}) = 0$ then the spatial cost is within the desired bound.

▶ **Lemma 15.** *The set function $-W$ is submodular.*

Proof. Since the sum of submodular functions is submodular, it is enough to show that $-\max\{0, \mathsf{cost}(i;\mathcal{C}) - r\} = \min\{0, -\mathsf{cost}(i;\mathcal{C}) + r\}$ is submodular. Thus it suffices to show that $-\mathsf{cost}(i;\mathcal{C})$ is submodular, and thus it suffices to show that $-d(p,\mathcal{C}(i))$, for all $p \in P(i)$, which is immediate since $d(p,\mathcal{C}(i)) = \min_{\tau\in\mathcal{C}} d(p,\tau(i))$. ◀

Algorithm. Our goal is to compute some set of trajectories $\mathcal{C}$ such that $W(\mathcal{C})$ is sufficiently small, while minimizing $|\mathcal{C}|$. The algorithm consists of the following steps:

Step 1. Let $\mathcal{C}_0$ be a set containing a single arbitrary trajectory.

Step 2. For any $i \in [L]$, let τ_i be a minimizer of $W(\mathcal{C}_{i-1} \cup \{\tau_i\})$. Set $\mathcal{C}_i = \mathcal{C}_{i-1} \cup \{\tau_i\}$.

Step 3. Return $\mathcal{C}_L$.

The parameter $L > 0$ will be determined later. The following Lemma bounds the running time of Step 2.

▶ **Lemma 16.** *Given a clustering $\mathcal{C}$, we can find τ minimizing $W(\mathcal{C}\cup\{\tau\})$, in time $\mathsf{poly}(|\mathcal{C}|, n)$.*

The above Lemma can be done via dynamic programming. The proof is essentially the same as in Lemma 13 and is thus omitted. We next show that for some value of L, the algorithm computes a low cost solution. To that end, we argue that with each iteration of the main loop, $W(\mathcal{C}_i)$ decreases significantly.

▶ **Lemma 17.** *If $\mathcal{I}$ admits a temporal (k, r, δ)-median-clustering, then for any $i \in \{1, \ldots, L\}$, there exists some feasible trajectory σ_i such that $W(\mathcal{C}_{i-1} \cup \{\sigma_i\}) \leq (1 - 1/k) \cdot W(\mathcal{C}_{i-1})$.*

Proof. Let $\mathcal{C}^* = \{\tau_1^*, \ldots, \tau_{k'}^*\}$ be a set of at most k trajectories that yields a (k, r, δ)-median temporal clustering. W.l.o.g. we may assume that $k' = k$. Let $K_0 = W(\mathcal{C}_{i-1})$, and for any $j \in [k]$, let $K_j = W(\mathcal{C}_{i-1} \cup \{\tau_1^*, \ldots, \tau_j^*\})$. Since $\mathcal{C}^*$ is a (k, r, δ)-median temporal clustering, it follows that $W(\mathcal{C}_{i-1}) = K_0 \geq K_1 \geq \ldots \geq K_k = 0$. For any $j \in [k]$, we also define $K'_j = W(\mathcal{C}_{i-1} \cup \{\tau_j^*\})$. By Lemma 15 we have that for all $j \in [k]$, $W(\mathcal{C}_{i-1}) - W(\mathcal{C}_{i-1} \cup \{\tau_j^*\}) \geq W(\mathcal{C}_{i-1} \cup \{\tau_1^*, \ldots, \tau_{j-1}^*\}) - W(\mathcal{C}_{i-1} \cup \{\tau_1^*, \ldots, \tau_j^*\})$. That is, $K_0 - K'_j \geq K_{j-1} - K_j$. Let $\ell = \arg\max_{j \in [k]} \{K_0 - K'_j\}$. It follows that $K_0 - K'_\ell = \max_{j \in [k]} \{K_0 - K'_j\} \geq \max_{j \in [k]} \{K_{j-1} - K_j\} \geq \frac{1}{k} \sum_{j=1}^{k} (K_{j-1} - K_j) = (K_0 - K_k)/k = K_0/k$. Let $\sigma_i = \tau_\ell^*$. It immediately follows that $W(\mathcal{C}_{i-1} \cup \{\sigma_i\}) = K'_\ell \leq (1 - 1/k) \cdot K_0 = (1 - 1/k) \cdot W(\mathcal{C}_{i-1})$, concluding the proof. ◀

We are now ready to prove Theorem 6.

Proof of Theorem 6. We first note that if $r = 0$, then a solution with k trajectories can be computed, if one exists, as follows: Since $r = 0$, it follows that every level of P has at most k points. We construct the flow network instance $N_\delta(P, P)$, as in Section 2.1. It is immediate that the flow instance is feasible iff there exists a solution with k trajectories. We may thus assume that $r > 0$. Since the minimum distance in M is 1, it follows that $r \geq 1$. In a generic step $1 \leq i \leq L$, let τ_i denote the trajectory returned by the dynamic program of Lemma 16, which minimizes $W(\mathcal{C} \cup \{\tau_i\})$. By Lemma 17, if $\mathcal{I}$ admits a temporal (k, r, δ)-median-clustering, then there exists some trajectory σ_i such that $W(\mathcal{C}_{i-1} \cup \{\sigma_i\}) \leq (1 - 1/k) \cdot W(\mathcal{C}_{i-1})$. Thus $W(\mathcal{C}_i) = W(\mathcal{C}_{i-1} \cup \{\tau_i\}) \leq W(\mathcal{C}_{i-1} \cup \{\sigma_i\}) \leq (1 - 1/k) \cdot W(\mathcal{C}_{i-1}) \leq (1 - 1/k)^i \cdot W(\mathcal{C}_0)$. Since the diameter of M is Δ, we get $W(\mathcal{C}_0) \leq \Delta \sum_{i \in [t]} |P(i)| = \Delta n$. Setting $L = k \ln(n\Delta/\varepsilon) = O(k \log(n\Delta/\varepsilon))$, we obtain $W(\mathcal{C}_L) \leq (1 - 1/k)^L n\Delta \leq \varepsilon \leq \varepsilon r$. Thus $\max_{i \in [t]} \max\{0, \mathsf{cost}(i; \mathcal{C}_L) - r\} \leq \sum_{i=1}^{t} \max\{0, \mathsf{cost}(i; \mathcal{C}_L) - r\} \leq \varepsilon r$, which implies $\mathsf{rad}_1(\mathcal{C}_L) = \max_{i \in [t]} \mathsf{cost}(i; \mathcal{C}) \leq (1 + \varepsilon) r$. It follows that either $\mathcal{C}_L$ is a $(L, 1 + \varepsilon, 1)$-approximation, or $\mathcal{I}$ does not admit a (k, r, δ)-median-clustering. Finally, the running time follows by the fact that we perform L iterations of the main loop; each in time bounded by Lemma 16. ◀

3 Inapproximability and Conclusion

We now briefly state our inapproximability results. Due to space constraints we defer all proofs to the full version [9]. There, we show that it is NP-hard to obtain a $(1, \mathsf{poly}(n), \mathsf{poly}(n))$-approximation, complementing Theorem 4.1. Further, we show that the problem remains hard to approximate, even for an inexact number of clusters where the points are taken from a nice metric space. Specifically, we prove that $(1.005, 2 - \varepsilon, \mathsf{poly}(n))$-approximation is NP-hard for points sampled from 2-dimensional Euclidean space. We show that the $(\ln n, 1, 1)$-approximation (Theorem 4.2) is best possible by observing that $((1-\varepsilon) \ln n, 2-\varepsilon', \cdot)$-approximation is NP-hard, though the construction involves a somewhat unnatural metric space. Finally, we adapt the hardness results for $(1, \mathsf{poly}(n), \mathsf{poly}(n))$-approximation and $(1.005, 2 - \varepsilon, \mathsf{poly}(n))$-approximation to Temporal k-Median/Means.

Conclusion. Our results show that many instances of temporal clustering are hard to approximate. On the other hand, our polynomial time approximations show that sometimes if we allow approximations in terms of parameters like r/δ or the spread Δ, the approximation becomes tractable. We wish to better understand the boundary between these cases. Another direction comes from altering the model; an alternative formulation could allow centers from the ambient metric space. We plan to investigate this model in future research.

References

1 Mohammad Ali Abam and Mark de Berg. Kinetic spanners in rd. In *Proceedings of the Twenty-fifth Annual Symposium on Computational Geometry*, SCG '09, pages 43–50, New York, NY, USA, 2009. ACM. URL: http://doi.acm.org/10.1145/1542362.1542371, doi:10.1145/1542362.1542371.

2 Pankaj K. Agarwal, Leonidas J. Guibas, Herbert Edelsbrunner, Jeff Erickson, Michael Isard, Sariel Har-Peled, John Hershberger, Christian S. Jensen, Lydia E. Kavraki, Patrice Koehl, Ming C. Lin, Dinesh Manocha, Dimitris N. Metaxas, Brian Mirtich, David M. Mount, S. Muthukrishnan, Dinesh K. Pai, Elisha Sacks, Jack Snoeyink, Subhash Suri, and Ouri Wolfson. Algorithmic issues in modeling motion. *ACM Comput. Surv.*, 34(4):550–572, 2002. URL: http://doi.acm.org/10.1145/592642.592647, doi:10.1145/592642.592647.

3 Alfred V. Aho and David Lee. Efficient algorithms for constructing testing sets, covering paths, and minimum flows. *AT&T Bell Laboratories Tech. Memo*, 159, 1987.

4 David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

5 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on computing*, 33(3):544–562, 2004.

6 Julien Basch, Leonidas J. Guibas, and John Hershberger. Data structures for mobile data. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '97, pages 747–756, Philadelphia, PA, USA, 1997. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=314161.314435.

7 Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, pcps, and nonapproximability—towards tight results. *SIAM J. Comput.*, 27(3):804–915, June 1998. URL: http://dx.doi.org/10.1137/S0097539796302531, doi:10.1137/S0097539796302531.

8 Sergio Cabello, Panos Giannopoulos, Christian Knauer, Dániel Marx, and Günter Rote. Geometric clustering: fixed-parameter tractability and lower bounds with respect to the dimension. *ACM Transactions on Algorithms (TALG)*, 7(4):43, 2011.

9 Tamal K. Dey, Alfred Rossi, and Anastasios Sidiropoulos. Temporal clustering. *CoRR*, abs/1704.05964, 2017. URL: http://arxiv.org/abs/1704.05964.

10 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 624–633, New York, NY, USA, 2014. ACM. doi:10.1145/2591796.2591884.

11 Anne Driemel, Amer Krivošija, and Christian Sohler. Clustering time series under the Fréachet distance. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 766–785. SIAM, 2016.

12 Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

13 Edward W Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–769, 1965.

14 Sorelle A. Friedler and David M. Mount. Approximation algorithm for the kinetic robust k-center problem. *Comput. Geom. Theory Appl.*, 43(6-7):572–586, August 2010. URL: http://dx.doi.org/10.1016/j.comgeo.2010.01.001, doi:10.1016/j.comgeo.2010.01.001.

15 Harold N. Gabow and Robert Endre Tarjan. Faster scaling algorithms for network problems. *SIAM J. Comput.*, 18(5):1013–1036, 1989. URL: https://doi.org/10.1137/0218069, doi:10.1137/0218069.

16 Jie Gao, Leonidas J. Guibas, and An Nguyen. Deformable spanners and applications. *Comput. Geom. Theory Appl.*, 35(1-2):2–19, August 2006. URL: http://dx.doi.org/10.1016/j.comgeo.2005.10.001, doi:10.1016/j.comgeo.2005.10.001.

17 Leonidas J. Guibas. Kinetic data structures: A state of the art report. In *Proceedings of the Third Workshop on the Algorithmic Foundations of Robotics on Robotics : The Algorithmic Perspective: The Algorithmic Perspective*, WAFR '98, pages 191–209, Natick, MA, USA, 1998. A. K. Peters, Ltd. URL: http://dl.acm.org/citation.cfm?id=298960.299007.

18 Sariel Har-Peled. Clustering motion. *Discrete & Computational Geometry*, 31(4):545–565, 2004.

19 Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, 2007. URL: http://dx.doi.org/10.1007/s00454-006-1271-x, doi:10.1007/s00454-006-1271-x.

20 Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300. ACM, 2004.

21 Sariel Har-Peled and Bardia Sadri. How fast is the k-means method? *Algorithmica*, 41(3):185–202, 2005.

22 Teresa W Haynes, Stephen Hedetniemi, and Peter Slater. *Fundamentals of domination in graphs*. CRC Press, 1998.

23 Dorit S Hochbaum and David B Shmoys. A best possible heuristic for the k-center problem. *Mathematics of operations research*, 10(2):180–184, 1985.

24 Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

25 David S Johnson. Approximation algorithms for combinatorial problems. *Journal of computer and system sciences*, 9(3):256–278, 1974.

26 Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. A local search approximation algorithm for k-means clustering. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 10–18. ACM, 2002.

27 Stavros G Kolliopoulos and Satish Rao. A nearly linear-time approximation scheme for the euclidean k-median problem. *SIAM Journal on Computing*, 37(3):757–782, 2007.

28 Madhukar R Korupolu, C Greg Plaxton, and Rajmohan Rajaraman. Analysis of a local search heuristic for facility location problems. *Journal of algorithms*, 37(1):146–188, 2000.

29 Stuart P Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

30 Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM (JACM)*, 41(5):960–981, 1994.

31 Mikkel Thorup. Quick k-median, k-center, and facility location for sparse graphs. In *Automata, Languages and Programming*, pages 249–260. Springer, 2001.