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.

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 [3, 7, 12, 18, 19, 20, 22, 23, 25, 26, 28, 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 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* [1, 5, 16, 17, 15, 13, 30]. 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 [9], 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.
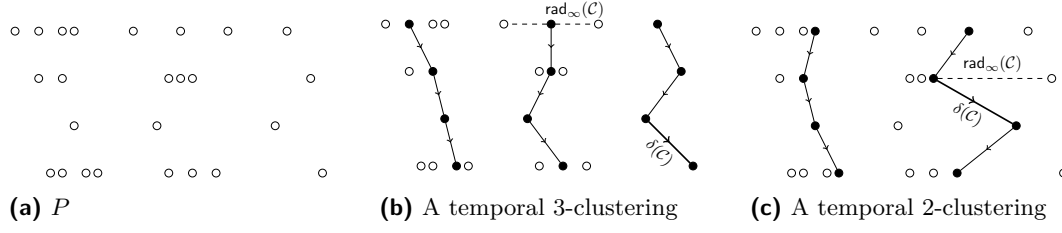
1.1 Problem formulations

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 $\mathsf{rad}_\infty(\mathcal{C}) = \max_{i \in [t]} \max_{p \in P(i)} d(p, \mathcal{C}(i))$, 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

(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.

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 [22]. 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 [21]. 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 [8, 29, 11]. 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.

We discuss Theorem 5.1 in Appendix B. The discussion of Theorem 5.2, and Theorem 5.3 are deferred to Appendix C, and Appendix D, respectively.

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 [4, 27]. In Section 2.4 we show that the local search approach fails in the
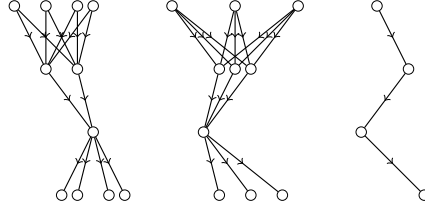
Figure 2 The graph $G_\delta(P)$ for P from the previous diagram and some δ. Points which are within a distance of δ in adjacent levels are connected by a directed edge which points toward the higher indexed level.

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 22. 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 in Appendix E and Theorem 7.2 in Appendix F 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.

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 $x \ne y$, 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 Appendix B complement this with an inapproximability result.

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 [2, 14]. 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.*

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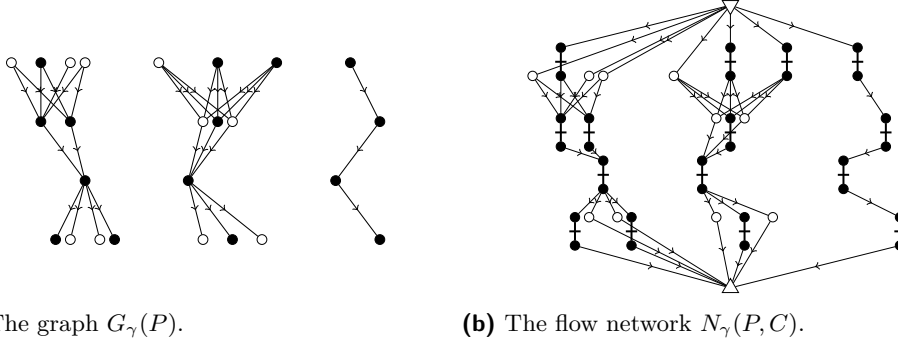
(a) The graph $G_\gamma(P)$. **(b)** The flow network $N_\gamma(P, C)$.

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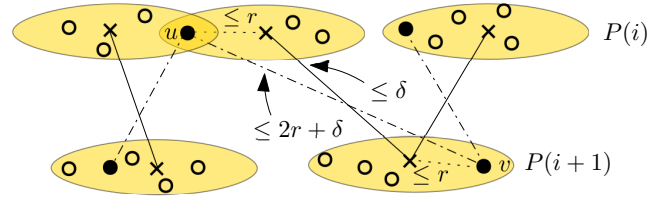
Figure 4 The crosses, filled circles, and empty circles are optimal centers, net points, and other points respectively. The paths (τ) in optimal solution and the deformed paths(τ') are indicated with solid and dotted edges respectively.

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 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 [14]. 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 Appendix C 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 Appendix A. 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 Appendix A. 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 [24]. 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 $k' = |\mathcal{S}'| \leq \ln n \cdot s_{\text{OPT}}$. By Lemma 12 it follows that $k' \leq \ln n \cdot k_{\text{OPT}} \leq \ln n \cdot k$. 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 later in Appendix D by showing that it is NP-hard to obtain a $(1.005, 2 - \varepsilon, \text{poly}(n))$-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 $\delta(\tau_j) \leq r$, 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 $N_{r+\delta}(P, C)$

(a)   **(b)**   **(c)**
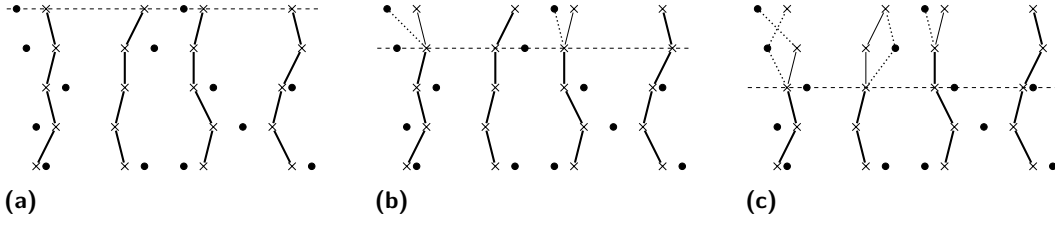
■ **Figure 5** An example of the inductive construction of the multisets of paths $\mathcal{Q}_i$, for $i = 0$ (Figure 5a), $i = 1$ (Figure 5b), and $i = 2$ (Figure 5c). Dotted lines show where a trajectory has been rounded to a net point. Thin and thick solid lines indicate where one or two trajectories 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.

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 [4, 27]. 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 $d(p, C(i))$ is replaced with the $d(p, C(i))^2$, and thus holds identically for TEMPORAL k-MEANS.

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

(a) The clustering $\mathcal{C}^*$.   **(b)** The clustering $\mathcal{C}$.
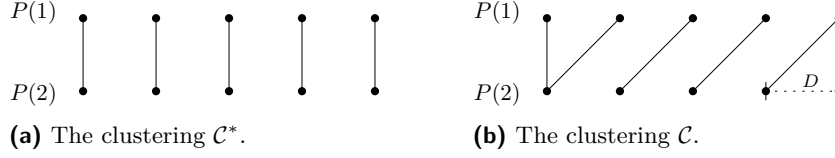
Figure 6 An example demonstrating that local search fails. Consider a temporal-sampling P of length 2 where $P(1) = P(2)$ consists of a sequence of 5 points where successive points are separated by D. (6a) A temporal $(5, r, \delta)$-median-clustering, $\mathcal{C}^*$, for any $r, \delta \in \mathbb{R}_{\geq 0}$ 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.

$M = (X, d)$. That is, $\Delta = \frac{\mathrm{diam}(M)}{\inf_{p,q \in X}\{d(p,q):d(p,q)>0\}}$. 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 $\mathsf{poly}(|\mathcal{C}|, n)$.*

Proof. This 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 the parameter L, the algorithm computes a solution with low cost. To that end, we show in the next Lemma that at each iteration of the main loop the quantity $W(\mathcal{C}_i)$ decreases by a significant amount.

▶ **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. For the remainder of the proof 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 we have that 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 the computed solution $\mathcal{C}_L$ is a $(L, 1 + \varepsilon, 1)$-approximation, or $\mathcal{I}$ does not admit a temporal (k, r, δ)-median-clustering, as required. Finally, the bound on the running time follows by the fact that we perform L iterations of the main loop; the running time of each iteration is 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 appendix. In Appendix B we show that temporal-clustering with an exact number of clusters is NP-hard to obtain a $(1, \mathsf{poly}(n), \mathsf{poly}(n))$-approximation (Theorem 5.1), complementing Theorem 4.1. In Appendix C we argue that the $(\ln n, 1, 1)$-approximation given in Theorem 4.2 is best possible by observing that $((1-\varepsilon)\ln n, 2-\varepsilon', \cdot)$-approximation is NP-hard (Theorem 5.2), though the construction involves a somewhat unnatural metric space. In Appendix D we show that $(1.005, 2 - \varepsilon, \mathsf{poly}(n))$-approximation is NP-hard even for points sampled from 2-dimensional Euclidean space (Theorem 5.3). In Appendix E (Theorem 7.1) and Appendix F (Theorem 7.2), we adapt the hardness results for $(1, \mathsf{poly}(n), \mathsf{poly}(n))$-approximation, and $(1.005, 2-\varepsilon, \mathsf{poly}(n))$-approximation to the TEMPORAL k-MEDIAN setting. We observe that these constructions 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.

Conclusion. Our results show that in many cases temporal clustering problems are hard to approximate. On the other hand, our polynomial time algorithms show that in some cases 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. We currently consider trajectories consisting of points in the input; an alternative formulation could allow centers from the ambient metric space. We plan to investigate this model in future research.

References

1 P. K. Agarwal, L. J. Guibas, H. Edelsbrunner, J. Erickson, M. Isard, S. Har-Peled, J. Hershberger, C. Jensen, L. Kavraki, P. Koehl, M. Lin, D. Manocha, D. Metaxas, B. Mirtich, D. Mount, S. Muthukrishnan, D. Pai, E. Sacks, J. Snoeyink, S. Suri, and O. Wolefson. Algorithmic issues in modeling motion. *ACM Comput. Surv.*, 34(4):550–572, December 2002.

2 A. V. Aho and D. Lee. Efficient algorithms for constructing testing sets, covering paths, and minimum flows. *Bell Lab. Tech. Memo*, 159, 1987.

3 D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proc. 18th Annu. ACM-SIAM Sympos. Disc. Alg.*, pages 1027–1035. SIAM, 2007.

4 V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location probs. *SIAM J. Comput.*, 33(3):544–562, 2004.

5 J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. In *Proc. 18th Annu. ACM-SIAM Sympos. Disc. Alg.*, SODA '97, pages 747–756. SIAM, 1997.

6 M. Bellare, O. Goldreich, and M. Sudan. Free bits, pcps, and nonapproximability—towards tight results. *SIAM J. Comput.*, 27(3):804–915, June 1998.

7 S. Cabello, P. Giannopoulos, C. Knauer, D. Marx, and G. Rote. Geometric clustering: fixed-parameter tractability and lower bounds with respect to the dimension. *ACM Trans. Algs. (TALG)*, 7(4):43, 2011.

8 I. Dinur and D. Steurer. Analytical approach to parallel repetition. In *Proc. 46th Annu. ACM Sympos. Thry. Comput.*, STOC '14, pages 624–633. ACM, 2014.

9 A. Driemel, A. Krivošija, and C. Sohler. Clustering time series under the Fréachet distance. In *Proc. 27th Annu. ACM-SIAM Sympos. Disc. Alg.*, pages 766–785. SIAM, 2016.

10 D. Eisenstat, P. N. Klein, and C. Mathieu. Approximating k-center in planar graphs. In *Proc. 25th Annu. ACM-SIAM Sympos. Disc. Alg.*, SODA '14, pages 617–627. SIAM, 2014.

11 U. Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

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

13 S. A. Friedler and D. M. Mount. Approximation algorithm for the kinetic robust k-center problem. *Comput. Geom. Thry. Appl.*, 43(6-7):572–586, August 2010.

14 H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for network problems. *SIAM J. Comput.*, 18(5):1013–1036, October 1989.

15 J. Gao, L. J. Guibas, and A. Nguyen. Deformable spanners and applications. *Comput. Geom.*, 35(1–2):2 – 19, 2006. Special Issue on the 20th ACM Sympos. Comput. Geom.

16 L. J. Guibas. Kinetic data structures: A state of the art report. In *Proc. 3rd Workshop on Alg. Found. of Robotics*, WAFR '98, pages 191–209. A. K. Peters, Ltd., 1998.

17 S. Har-Peled. Clustering motion. *Disc. & Comput. Geom.*, 31(4):545–565, 2004.

18 S. Har-Peled and A. Kushal. Smaller coresets for k-median and k-means clustering. *Disc. & Comput. Geom.*, 37(1):3–19, 2007.

19 S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *Proc. 36th Annu. ACM Sympos. Thry. Comput.*, pages 291–300. ACM, 2004.

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

21 T. W. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of domination in graphs*. CRC Press, 1998.

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

23 A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

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

25   T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. A local search approximation algorithm for k-means clustering. In *SoCG 18*, pages 10–18. ACM, 2002.

26   S. G. Kolliopoulos and S. Rao. A nearly linear-time approximation scheme for the euclidean k-median problem. *SIAM J. Comput.*, 37(3):757–782, 2007.

27   M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. *J. Algorithms*, 37(1):146–188, 2000.

28   S. P. Lloyd. Least squares quantization in pcm. *IEEE Trns. Inf. Thy.*, 28(2):129–137, 1982.

29   C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994.

30   M. de Berg M. Ali Abam. Kinetic spanners in rd. In *SoCG 25*, pages 43–50. ACM, 2009.

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

Proofs of Lemmas for $(\ln(n),1,1)$**-approximation**

Proof of Lemma 12. Let $\mathcal{C} = \{c_i\}_{i=1}^{t}$ be a temporal (k',r,δ)-clustering of P, for some $k' \in \mathbb{N}$. Let $\tau^1, \ldots, \tau^{k'}$ be the natural decomposition of $\mathcal{C}$ into a collection of trajectories, where $\tau^j = c_1(j), \ldots, c_t(j)$, for all $i \in [k']$. Let $\mathcal{S}' = \bigcup_{j \in [k']}\{\mathsf{tube}(\tau^j, r)\}$. Since $\delta(\mathcal{C}) \le \delta$, it follows that $\delta(\tau^j) \le \delta$, for all $j \in [k']$. Thus, $\tau^j \in \mathcal{T}_\delta(P)$, for all $j \in [k']$, which implies that $\mathcal{S}' \subseteq \mathcal{S}$. Since $\mathcal{C}$ is a temporal (k',r,δ)-clustering, it follows that for all $i \in [t]$, for all $x \in P(i)$, there exists some $j \in [k']$ such that $d(\tau^j(i), x) \le r$. It follows that

$$U = \bigcup_{i \in [t]} V(P, i) \subseteq \bigcup_{j \in [k']} \mathsf{tube}(\tau^j, r) = \bigcup_{X \in \mathcal{S}'} X.$$

We have established that $\mathcal{S}'$ is a valid solution for $(U, \mathcal{S}')$, with $|\mathcal{S}'| = k'$, which implies that $s_{\mathrm{OPT}} \le k_{\mathrm{OPT}}$.

Conversely, let $\mathcal{S}' \subseteq \mathcal{S}$ be a solution for $(U, \mathcal{S})$. Let $I = \{\tau \in \mathcal{T}_\delta(P) : \{\mathsf{tube}(\tau, r)\} \in \mathcal{S}'\}$. Fix an arbitrary ordering $I = \{\tau^1, \ldots, \tau^{|\mathcal{S}'|}\}$. We may now define the clustering $\mathcal{C} = \{c_i\}_{i \in [t]}$, where $c_i(j) = \tau^j(i)$, for all $i \in [t]$, $j \in [|\mathcal{S}'|]$. Since $\mathcal{S}'$ is a feasible solution for $(U, \mathcal{S})$, it is immediate that $\mathcal{S}$ is a temporal $(|\mathcal{S}'|,r,\delta)$-clustering, which implies that $k_{\mathrm{OPT}} \le s_{\mathrm{OPT}}$.   ◄

Proof of Lemma 13. We first establish some notation. Let $G = G_\delta(P)$. Note that due to the orientation of the edges, any path Q in G has at most one vertex at level i for any $i \in [t]$. Recall that this vertex is of the form (i, x) for some $x \in P(i)$. For convenience we let $Q(i) = x$ whenever such a vertex exists. Otherwise, we say that $Q(i) = \mathsf{nil}$. We extend the notion of a tube to paths in G, by defining

$$\mathsf{tube}(Q, r) = \bigcup_{i \in [t] : Q(i) \ne \mathsf{nil}} \{(i, x) \in V(P, i) \mid x \in \mathsf{ball}(Q(i), r)\}$$

We also denote those elements of U which are not covered by $\mathcal{S}'$ as $\mathsf{uncovered}(\mathsf{U}, \mathcal{S}')$. In other words, $\mathsf{uncovered}(\mathsf{U}, \mathcal{S}') = U \setminus \bigcup_{Y \in \mathcal{S}'} Y$.

The algorithm computes the desired X via dynamic programming, as follows. The dynamic programming table is indexed by U. For each $(i, x) \in U$, we compute a path Q in G that we store at location (i, x) of the table. More precisely, if there is no path from (i, x) to some vertex in (t, y), where $y \in P(t)$, then we set $Q = \mathsf{nil}$. Otherwise, we set Q to be a path that starts from (i, x), and terminates at some vertex (t, y), with $y \in P(t)$, maximizing the quantity $\mathsf{val}(Q) = |\mathsf{tube}(Q, r) \cap \mathsf{uncovered}(\mathsf{U}, \mathcal{S}')|$. For every $x \in P(t)$, the only choice for Q is $Q = x$. Thus we may fill in the entries of the table indexed by (t, x), for all $x \in P(t)$. Next, for each $i \in [t-1]$, for each $x \in P(i)$, we compute the path Q to be stored at location (i, x), assuming that all the entries indexed by $(i+1, y)$, for all $y \in P(i+1)$ have already been computed. It is immediate for each Q that starts from x and terminates at $P(t)$, we have $\mathsf{val}(Q) = |\mathsf{ball}(x, r) \cap \mathsf{uncovered}(\mathsf{U}, \mathcal{S}')| + \mathsf{val}(Q')$, where Q' is the suffix of Q obtained after removing x. Thus, in order to compute the desired path Q for x that maximizes $\mathsf{val}(Q)$, it suffices to find the path Q' that maximizes $\mathsf{val}(Q')$, and starts from some neighbor of x in $P(i+1)$. This completes the description of the algorithm for filling in the values of the dynamic table. We may now set $X = \mathsf{tube}(Q^*, r)$, where Q^* is the path stored in the entry $(1, x)$, for some x that maximizes $\mathsf{val}(Q^*)$. We now show how to complete this procedure in $O(n^2)$ time. During a precomputation phase we can construct $\mathsf{uncovered}(\mathsf{U}, \mathcal{S}')$ in $O(n)$ time. Further, we can check whether an uncovered point intersects the $\mathsf{ball}(x, r)$ in constant-time. This allows us to precompute and store $|\mathsf{ball}(x, r) \cap \mathsf{uncovered}(\mathsf{U}, \mathcal{S}')|$ for each node $(i, x) \in V(G)$ in time linear time, for a total of $O(n^2)$ over all nodes. Now note that

in populating the table, the algorithm visits each node in $V(G)$ and evaluates the choice of taking the path associated with each of its successors in G as a suffix. By also keeping the value of a path in the table this decision can be made in constant time using only stored or precomputed information. Thus the algorithm takes $O(n^2)$ time over all.   ◀

B   Inapproximability with exact number of clusters

We complement Theorem 4.1 by showing that it is NP-hard to obtain a $(1, \mathsf{poly}(n), \mathsf{poly}(n))$-approximation. Further, this inapproximability result holds even for a temporal sampling in $\mathbb{R}^2$. Let P be such a sample consisting of n points. We show that given any fixed $\varepsilon > 0$ and $s \in [0, 1]$ there exists universal constants c_r, c_δ such that the $(1, c_r n^{s(1-\varepsilon)}, c_\delta n^{(1-s)(1-\varepsilon)})$-approximation problem is NP-hard. We describe a polynomial-time reduction from instances of 3-SAT to temporal-samplings over $\mathbb{R}^2$. In particular, given any positive real numbers ε, $s \leq 1$, r_0, and $\delta_0 < r_0\sqrt{3}/4$, and some l-variable instance $\mathcal{S}$ of 3-SAT, we construct a temporal sampling P such that the following conditions hold:

1. P admits a temporal (l, r_0, δ_0)-clustering if $\mathcal{S}$ is satisfiable.
2. P does not admit a temporal $(l, c_r n^{s(1-\varepsilon)} r_0, c_\delta n^{(1-s)(1-\varepsilon)} \delta_0)$-clustering otherwise.

Suppose we are given an instance $\mathcal{S}$ of 3-SAT with l variables and m clauses. W.l.o.g., we assume that every clause contains no repeated variables. We now describe how to produce the corresponding temporal-sampling P:
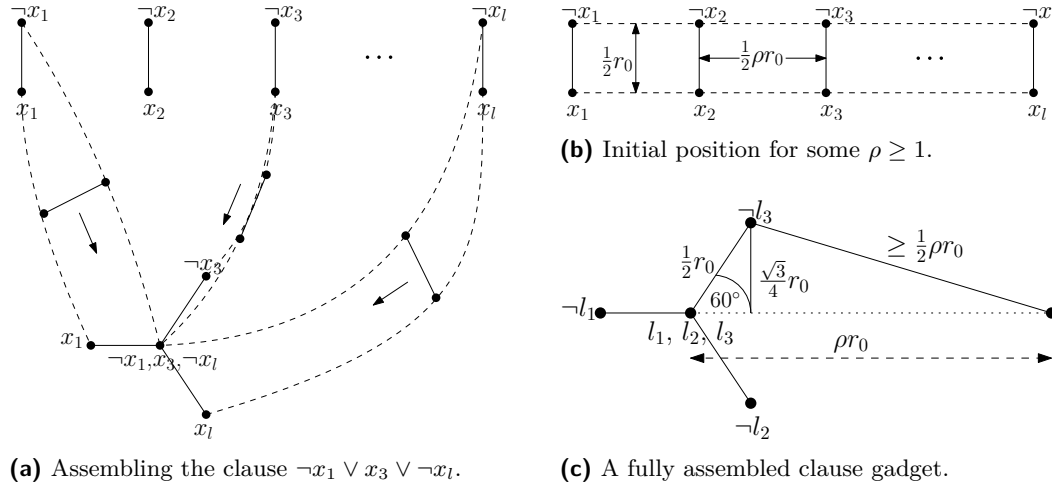
Variable gadgets. For each variable x_i of $\mathcal{S}$ we introduce a pair of points in each level of P. We denote these points by x_i and $\neg x_i$ and the pair $\{x_i, \neg x_i\}$ by v_i. We will sometimes refer to these points as literals. Initially, we lay out the variable gadgets in the plane such that $d(x_i, \neg x_i) = \frac{1}{2}r_0$ for all $1 \leq i \leq l$, and $\rho r_0/2 \leq d(v_i, v_j) \leq l\rho r_0/2$ for all $1 \leq i < j \leq l$. Here, $\rho = \rho(l) \geq 1$ denotes a yet to be determined function of the number of variables (see Figure 7b). We refer to this configuration as *initial position*.

Clause gadgets. Order the clauses of $\mathcal{S}$ arbitrarily as $c_1, \ldots, c_m$. For each clause we build a series of levels for the temporal-sampling where the variable gadgets of that clause appear to undergo rigid motion see (Figure 7a). By *motion* we mean that points in any pair of subsequent levels which correspond to the same literal are within a bounded distance of each other. Further, we say it is *rigid* because in every level we maintain the distance of $\frac{1}{2}r_0$ between literals of the same variable gadget. Note that we may enforce a consistent assignment of a center to a literal between consecutive levels by ensuring that the distance between a literal and its copy in the next level is at most δ_0, and that the distance between points corresponding to distinct literals exceeds δ_0. We describe this motion in three phases:

Phase 1: Assembly. All points start in the initial position (see Figure 7b). The variable gadgets which are used in the specified clause are brought together one by one under rigid motion (see Figure 7a). This motion brings the ends of the variable gadgets which correspond to the literals that appear in the clause to a single common point. The unused literals appear on a circle of radius $\frac{1}{2}r_0$ about the common point, arranged such that they form an equilateral triangle (see Figure 7c). This phase takes $O(l\rho r_0/\delta_0)$ steps per clause.

Phase 2: Satisfiability check. An extra point is introduced at the location where the variable gadgets meet. This point then undergoes motion directly away from one of the unused literals for a distance of ρr_0, before reversing course and returning to the common point at the center of the gadget (see Figure 7c). It subsequently disappears. This phase takes $O(\rho r_0/\delta_0)$ steps per clause.

Phase 3: Disassembly. The motion of *Phase 1* is reversed and the variable gadgets are returned to initial position in $O(l\rho r_0/\delta_0)$ steps per clause.

(a) Assembling the clause $\neg x_1 \vee x_3 \vee \neg x_l$.

(b) Initial position for some $\rho \geq 1$.

(c) A fully assembled clause gadget.

■ **Figure 7** Reference diagrams for gadget construction.

Analysis. Let P denote the temporal-sampling of $\mathbb{R}^2$ given by the above construction, and let r_{OPT} denote the smallest value of r such that P admits a temporal (l,r,δ_0)-clustering.

▶ **Lemma 18.** *If $\mathcal{S}$ is satisfiable then $r_{\text{OPT}} \leq r_0$.*

Proof. Since $\mathcal{S}$ is satisfiable there exists a satisfying assignment. We now exhibit a solution of cost at most r_0. For each of the l variable gadgets, we set one of the two points to be a center in the initial configuration. If x_i is TRUE in the satisfying assignment then the point which corresponds to the literal x_i is selected as a center. Otherwise, $\neg x_i$ is selected. Note that the construction of the gadget ensures that the same choice of literals can be maintained throughout the entire motion. We maintain these choices during *Phase 1* and *Phase 3* where the only points which appear are from variable gadgets. Since at least one side of each variable gadget is covered and the distance between the points of these gadgets are $\frac{1}{2}r_0$ at all times, the covering cost of each level in these phases is $\frac{1}{2}r_0$. By satisfiability there is at least one center at the common point where the clause literals meet. We take one such center and assign it to the extra point. Throughout *Phase 2* the extra point is covered and the clause gadget retains at least one center (somewhere) at a coverage cost of at most r_0. ◀

▶ **Lemma 19.** *If $\mathcal{S}$ is not satisfiable then $r_{\text{OPT}} \geq \frac{1}{2}\rho r_0$.*

Proof. We will show that a satisfying assignment can be inferred from a clustering $\mathcal{C}$ with cost below $\frac{1}{2}\rho r_0$. First, we argue that the variable gadgets are consistent in initial position. Since $\mathsf{rad}_\infty(\mathcal{C}) < \frac{1}{2}\rho r_0$, it follows that whenever the point configuration is in initial position, every variable gadget has exactly one center. Thus we do not simultaneously select a literal and its negation, as otherwise it follows by the pigeonhole principle that at least one variable gadget is uncovered and $\mathsf{rad}_\infty(\mathcal{C}) \geq \rho r_0$. Next, we argue that these choices must remain consistent within a clause gadget $c = (l_1 \vee l_2 \vee l_3)$. The only opportunity for a trajectory to change literals is when two literals are within a distance of δ_0, and the only literals in the gadget which pass within δ_0 are l_1, l_2, and l_3. To see that inconsistency is expensive, assume that l_1 is a center in the first level of the gadget but not in the last, then the variable gadget which contains literal l_1 is uncovered in the last level for the clause, implying $\mathsf{rad}_\infty(\mathcal{C}) \geq \rho r_0$. Moreover, since $\delta_0 < r_0$ the choice of cluster centers at the end of clause gadget c_i must be

the same as at the beginning of c_{i+1} for $1 \leq i < t$. We now argue that at least one of the literals for a clause is a center. Suppose none of them are a center, then the extra point from *Phase 2* never coincides with a center as $\delta_0 < r_0\sqrt{3}/4$ and the closest possible location for a center to any point along its motion is at a distance of $r_0\sqrt{3}/4$ away. At its maximum displacement the extra point is at a distance of ρr_0 from where the literals meet, which is at least $\frac{1}{2}\rho r_0$ from any other center. Thus, $\mathsf{rad}_\infty(\mathcal{C}) \geq \frac{1}{2}\rho r_0$. We produce a satisfying assignment by setting true those literals which are assigned centers in $\mathcal{C}$.   ◀

▶ **Remark.** We remark that Lemma 18 and Lemma 19 continue to hold even for $\delta = \beta\delta_0$ for any $1 \leq \beta < (\sqrt{3}/4)r_0/\delta_0$. Essentially, there are two important distance scales given by δ_0 and r_0. Points in successive levels closer than $\beta\delta_0$ can be assumed to be δ_0 and vice versa.

▶ **Lemma 20.** *Given an instance $\mathcal{S}$ of* 3-SAT *with l variables and m clauses. It is possible to construct the above temporal-sampling of size $n \in O(\rho r_0 l^2 m/\delta_0)$ in $O(\rho r_0 l^2 m/\delta_0)$-time.*

Proof. This is immediate by construction, as each clause gadget consists of $O(\rho r_0 l/\delta_0)$ levels of size $O(l)$ and there are m clauses.   ◀

Proof of Theorem 5.1. Let $c > 5(1/\varepsilon - 1)$. Let $\mathcal{S}$ be an instance of 3-SAT with l variables and m clauses. We invoke Lemma 20 with $\rho(l) = l^{s \cdot c}$, $r_0(l)/\delta_0(l) = l^{(1-s) \cdot c}$ and to yield, in polynomial-time, a temporal-sampling P over $\mathbb{R}^2$ of size $n \in O(l^{c+5})$. Note here we have used the fact that $m \in O(l^3)$. Suppose the existence of some polynomial-time $(1, \alpha(n), \beta(n))$-approximation where $\alpha(n) = c_r n^{s \cdot (1-\varepsilon)}$, and $\beta(n) = c_\delta n^{(1-s)(1-\varepsilon)}$. There exists a universal constant c_1 (where $c_1 > 1$ by construction) such that $c_1 n^{(1-s) \cdot (1-\varepsilon)} \leq (\sqrt{3}/4)r_0/\delta_0 = (\sqrt{3}/4)l^{(1-s) \cdot c}$. Thus we let $1 \leq c_\delta < c_1$, Remark B indicates that Lemma 18 and Lemma 19 still apply. We run this approximation on P with $k = l$, $r = r_0$, and $\delta = \delta_0$. In polynomial-time the algorithm either produces output or fails. If the algorithm fails it follows by Definition 3 that P does not admit a temporal (l, r_0, δ_0)-clustering and thus $\mathcal{S}$ is not satisfiable by Lemma 18. Otherwise, the algorithm produces some clustering $\mathcal{C}$ of radius $\mathsf{rad}_\infty(\mathcal{C}) \leq \alpha(n)r_0$. Lemma 19 ensures that if $\mathcal{S}$ is not satisfiable then $\mathsf{rad}_\infty(\mathcal{C}) \geq \frac{1}{2}\rho r_0 = \frac{1}{2}l^{s \cdot c}r_0 \geq c_0 n^{sc/(c+5)}r_0 \geq c_0 n^{s \cdot (1-\varepsilon)}r_0$ for some universal constant $c_0 > 0$. It follows that if $c_r \in (0, c_0)$ the algorithm produces output if and only if $\mathcal{S}$ is satisfiable, giving a polynomial-time test for satisfiability.   ◀

C   NP-**hardness of** $((1-\varepsilon)\ln(n), 2 - \varepsilon', 1)$-**approximation**

We now argue that the $(\ln(n), 1, 1)$-approximation is tight. The r-DOMINATING SET problem for a metric space (X, d) is to find a smallest set of points C such that every point in X is at most a distance r from a point in C. It is known that SET-COVER reduces to r-DOMINATING SET in polynomial-time [10]. For completeness we now give such a reduction:

▶ **Folklore 21.** SET-COVER reduces to r-DOMINATING SET in polynomial-time. Moreover, any $\alpha(n)$-approximate solver for r-DOMINATING SET yields an $\alpha(n)$-approximation for SET-COVER.

Proof. Let $(U, \mathcal{S})$ be an instance of SET-COVER, where $\mathcal{S} = \{S_1, \ldots, S_n\}$. We define a metric space $M = (X, d)$ where X contains points corresponding to the elements of U, and $\mathcal{S}$, and where d satisfies the following:

1. $d(S_i, S_j) = 1$ for all $S_i, S_j \in \mathcal{S}$, $i \neq j$.
2. $d(u, S_i) = 1$ for any $S_i \in \mathcal{S}$, u such that $u \in S_i$.
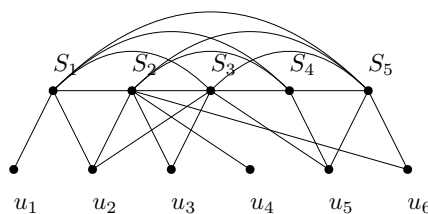
Figure 8 The r-Dominating Set instance corresponding to the Set-Cover instance $(U, \mathcal{S})$ with $U = \{u_1, \ldots, u_6\}$ and $S = \{S_1, \ldots, S_5\}$, where $S_1 = \{u_1, u_2\}$, $S_2 = \{u_2, u_3, u_4, u_6\}$, $S_3 = \{u_2, u_3, u_5\}$, $S_4 = \{u_5\}$, $S_5 = \{u_5, u_6\}$. Edges are shown between points at distance of 1. All other pairs of distinct points have a distance of 2.

3. Otherwise, $d(u, v) = 2$ for any remaining pair of distinct points $u, v \in X$.

These constraints on d, together with the requirement that it be a metric on M, completely determine d. (See Figure 8 for an illustration.)

Let S be a feasible solution to the Set-Cover instance. Note that S induces a 1-Dominating Set solution of size $|S|$ by taking the set of points in X which correspond to the points in S. On the other hand, any feasible solution of 1-Dominating Set can be converted into a feasible solution of Set-Cover in linear time without increasing its size. To see this fix a feasible solution $R' \subset X$, and let $u \in R' \cap U$. Note that a 1-ball of u in M consists only of u and points for elements of $\mathcal{S}$ which cover it. Instead we can cover u by any other point in its 1-ball, S_i. Since $\mathsf{ball}(u, 1) \subseteq \mathsf{ball}(S_i, 1)$ the feasibility of the solution is preserved. Performing this replacement for all points in $R' \cap U$ induces a feasible solution to Set-Cover of size no more than $|R'|$. It follows that any $\alpha(n)$-approximate solver for r-Dominating Set yields an $\alpha(n)$-approximation for Set-Cover. ◀

We now prove Theorem 5.2.

Proof of Theorem 5.2. As a corollary to Folklore 21, any polynomial-time $((1 - \varepsilon) \ln(n), 2 - \varepsilon', \cdot)$-approximation for r-Dominating Set yields a polynomial-time $((1 - \varepsilon) \ln(n))$-approximation for Set-Cover problem, by taking P to be a single level temporal-sampling with $P(1) = C$ and invoking it with $r = 1$, $\delta = 0$, and successive values of k until it succeeds in producing a clustering. Since the first non-failing value of k is at most the size of an optimum solution to Set-Cover, the resulting clustering is at most $(1 - \varepsilon) \ln(n)$ times larger. The hardness now follows for any $\varepsilon > 0$ by a result of Dinur and Steurer [8]. ◀

D   Inapproximability in 2-dimensional Euclidean space

While Theorem 5.2 shows that an increase in the number of clusters should be expected if we demand to have a polynomial-time algorithm that closely approximates the optimal radius and displacement, the construction involves a somewhat unnatural metric space. We show that this condition remains even for 2-dimensional Euclidean space.

▶ **Theorem 22** (Hardness of MAX-2-SAT[6]). *There exist constants $0 < s < c < 1$, satisfying $c > 0.9$ and $c/s = 74/73$, such that it is* NP-*hard to decide whether a given* 2-CNF *formula admits an assignment which satisfies at least c-fraction of the clauses, or whether any assignment satisfies at most s-fraction of clauses.*

Let $\mathcal{S}$ be an instance of MAX-2-SAT consisting of l variables and m clauses. Given $0 < \delta_0 < r_0$, our goal is to construct a temporal-sampling P in polynomial-time such that:

1. P admits a temporal $(2m + (1 - c)m, r_0, \delta_0)$-clustering if there exists a truth assignment that satisfies at least $c \cdot m$ clauses of $\mathcal{S}$.
2. P does not admit a temporal (k, ρ, δ_0)-clustering for any $k < 2m + (1 - s)m$ and $\rho < 2r_0$, if every truth assignment satisfies at most $s \cdot m$ clauses,

where c and s are the constants from Theorem 22

Variable gadgets.

For each variable x_i of $\mathcal{S}$, let k_i be the number of literals where it appears. We introduce k_i pairs of points into each level of P, which we denote by x_i^j and $\neg x_i^j$ for $j \in [k_i]$. Our initial arrangement of variable gadgets in the first level of P is as in the top row of Figure 9.

Clause gadgets.

We arrange a configuration of points for each clause $c = (l_1 \vee l_2)$ of $\mathcal{S}$ by rigidly transporting one of the variable gadget copies of each of its variables to a predetermined location. In doing so we overlay them so that the points of each gadget corresponding to l_1 and l_2 overlap on one side (see Figure 9g). The distance between two neighboring clause gadgets is $4r_0$.

Phase 1: Consistency checking and clause assembly. Each variable in $\mathcal{S}$ corresponds to one or more variable gadgets (which we think of as copies). We check each pair of variable gadget copies to ensure that all copies have at least one side selected as a center. That is, either x_i^j has a center for all $j \in [k_i]$, or all $\neg x_i^j$ do (or both). Initially, we declare all variable gadget copies corresponding to the same variable as "unchecked". For each variable we perform the following procedure: The unchecked copy with smallest index j (the left most one, see Figure 9) undergoes rigid motion so that it is aligned end to end with a higher indexed copy $j' > j$. The two copies align with a distance of $2r_0$ in-between. Moreover, the gadgets are consistently oriented with $\neg x_i$ on the left and x_i on the right. A consistency check is performed between the pair consisting of the following steps: 1. An additional point is introduced for one level at the mid-way point between the two gadgets. 2. It subsequently disappears and the two variable gadgets simultaneously rotate in place in the same direction about their respective midpoints so that they are both oriented in with x_i on the left. 3. The additional point reappears for a single level. 4. After this the variable gadget copies rotate once more in place so that x_i is on the right. The j-th copy now undergoes rigid motion to check against the next copy ($j' + 1$-st, if it exists). After all j has checked with all $j' > j$, it is declared "checked" and undergoes rigid motion to take its place in the clause gadget. This process repeats until all copies of the variable are checked. Finally, the lone remaining copy undergoes rigid motion to its place in some clause gadget. The total traveled distance for any copy of a variable gadget is $O(mr_0)$. Since it may only do so in steps of size δ_0, the total number of steps per copy is $O(mr_0/\delta_0)$ for a total of $O(m^2 r_0/\delta_0)$ over all variable copies.

Phase 2: Satisfiability check. An extra point is introduced in each clause gadget. For each clause $c = (l_1 \vee l_2)$, this occurs at the point where the sides of the variable gadget copies corresponding to l_1 and l_2 have been identified. These points undergo motion along a line orthogonal to the variable gadget copies for a distance of $4r_0$ (see Figure 9h). This phase takes $O(r_0/\delta_0)$ steps.

▶ **Lemma 23.** *If there exists a truth assignment satisfying at least $c \cdot m$ clauses then P admits a temporal $(2m + (1 - c)m, r_0, \delta_0)$-clustering.*

Proof. Suppose such a truth assignment, T, exists. We use this assignment to determine $2m$ of the $2m + (1 - c)m$ cluster centers. For each variable x_i, if $T(x_i)$ is TRUE we select the points x_i^j for all $j \in [k_i]$ as centers in all levels up to and including the first *Phase 2* level. Otherwise we select all $\neg x_i^j$ for all $j \in [k_i]$ as centers in those same levels. With these selections each satisfied clause has at least one center present at the extra point of the clause gadget at the start of *Phase 2*. Hence, for satisfied clauses we arbitrarily select one of these centers and assign it to the extra point for all remaining levels. For the unsatisfied clauses we assign one of the remaining $(1 - c)m$ centers. Note that the ends of the variable gadgets that meet at the extra point of each unsatisfied clause correspond to (thus far) unselected literals. We select one of those literals from each clause to be a center in all levels up to and including the first *Phase 2* level, and select each unsatisfied clause's extra point in the remaining levels. The remaining centers can be assigned arbitrarily to some x_i^j in all levels. Let the clustering induced by this choice of centers be denoted as $\mathcal{C}$.

We now argue that $\mathsf{rad}_\infty(\mathcal{C}) \le r_0$. First note that for each variable, x_i, for any $j \in [k_i]$ either the point x_i^j or $\neg x_i^j$ is a center by assignment from T. Since the variable gadgets only undergo rigid motion, any level of P with only the points from the variable gadgets is covered at a cost of at most r_0. We need only consider the levels of P where additional points exist. Since the assignment from T is consistent, it follows that the extra points introduced during the consistency check involving x_i^j and x_i^{j+1} for $j \in [k_i - 1]$ are always within a distance of r_0 of their corresponding centers. For the levels in *Phase 2* all extra points are covered and at least one center remains on each clause gadget, which has diameter r_0. ◄

▶ **Lemma 24.** *If every truth assignment satisfies at most $s \cdot m$ clauses then P does not admit a temporal (k, ρ, δ_0)-clustering for $k < 2m + (1 - s)m$, $\rho < 2r_0$*

Proof. We argue that if P admits temporal (k, ρ, δ_0)-clustering for some $k < 2m + (1 - s)m$, $\rho < 2r_0$, then there exists a truth assignment that satisfies more than $s \cdot m$ clauses. For any variable, all copies of its gadget in the first level contain at least one center, as otherwise one of them is completely uncovered and the nearest center is at least a distance of $2r_0$ away. Note that the only points of a variable gadget which pass within a distance of another δ_0 are points which are eventually identified within a clause gadget (just before *Phase 2*). Thus, the choice of centers in the first level completely determines the choice of centers at the start of *Phase 2*. Further, this assignment is consistent in the sense that for each variable x_i either x_i^j for all $j \in [k_i]$ are centers, or all of $\neg x_i^j$ for all $j \in [k_i]$ are (this is not mutually exclusive, some variable gadgets might have both). To see why suppose that for some variable x_i there is a pair of copies corresponding to $j, j' \in [k_i - 1]$ which disagree. In this case each copy has exactly one center. Without loss of generality suppose x_i^j and $\neg x_i^{j'}$ have centers. There is a level where the extra point of the consistency check is exactly midway between the opposite ends of these copies. That is, the extra point is midway between $\neg x_i^j$ and $x_i^{j'}$. It follows that the nearest centers to the extra point are at a distance of $2r_0$ contradicting that $\rho < 2r_0$. Finally, note that the extra point of each clause gadget is covered. This is only possible if at least one of the literals that meet at the extra point have a center. Now consider the truth assignment T, where $T(x_i)$ is TRUE if and only if all x_i^j for all $j \in [k_i]$ are centers in *Phase 1*. Let $|T|$ denote the number of clauses satisfied by T. We know that this assignment together with $u = k - 2m$ additional centers is sufficient to satisfy all clauses. That is, $|T| + u >= m$. Since there are only $k < 2m + (1 - s)m$ centers in total, it follows that the number of additional centers required is strictly less than $(1 - s)m$. Thus, $(1 - s)m + |T| > |T| + u >= m$, and we conclude that $|T| > s \cdot m$. ◄
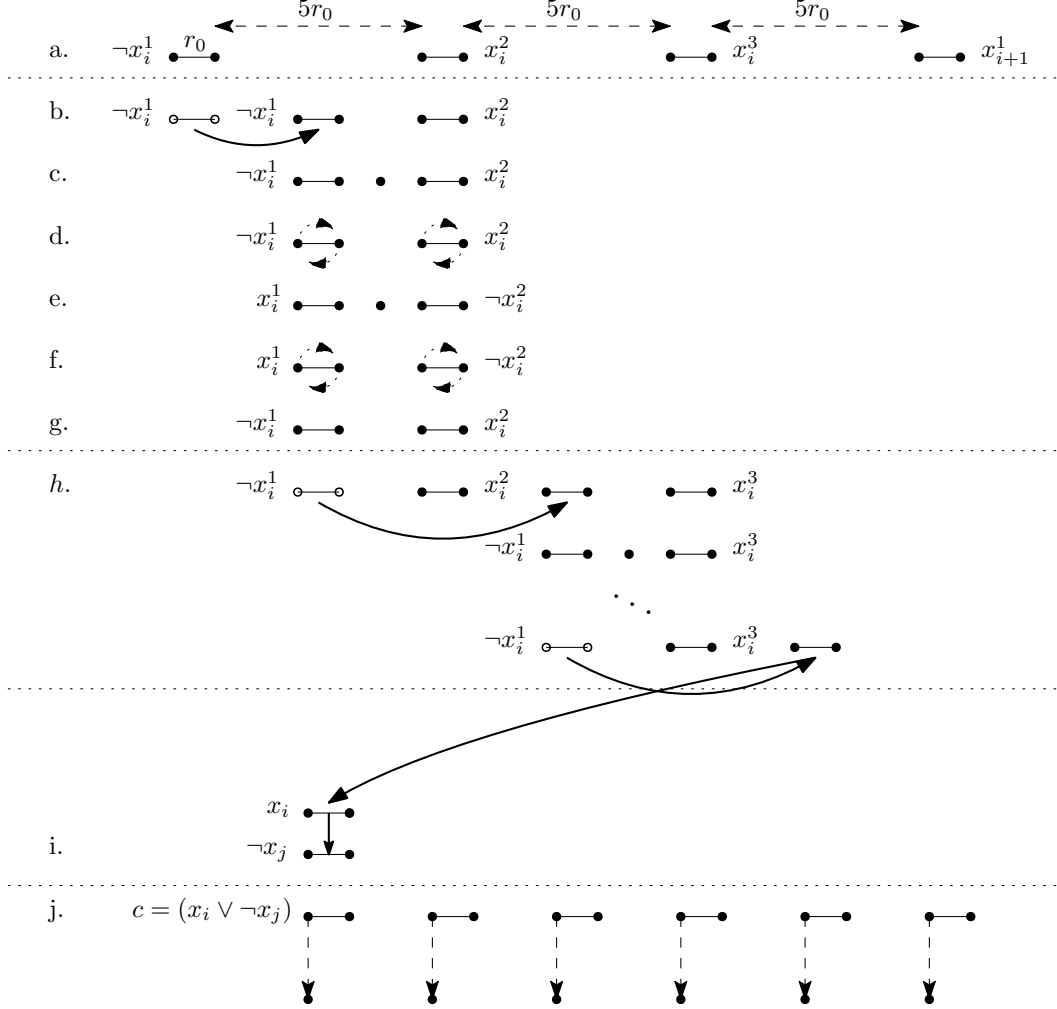
Figure 9 *Schematic overview of Phases 1 and 2.* Line a shows an example of the initial position of variable gadget copies corresponding to variable x_i and the first variable gadget copy of x_{i+1}. Lines b through g show the consistency checking of *Phase 1* as time progresses (downwards). In line b, the variable gadget copy x_i^1 moves to align end-to-end with x_i^2. Next an extra point appears mid-way between them for a single level (line c). It disappears and then the gadgets rotate so that they are flipped over (line d). An extra point then reappears (line e). In line f, both gadgets flip back over, ending in the position on line g. In line h, this process continues with x_i^1 moving to align with x_i^3, before again performing the consistency check against x_i^3. This process goes on repeating until x_i^1 has checked against x_i^j for all $j > 1$. After this x_1 goes on to assemble a clause $c = (x_i \vee \neg x_j)$ in line i by overlapping with a copy of a variable gadget for x_j (already in place). The process then repeats with x_i^2 checking against x_i^j for $j > 2$. After all variable gadget copies have been checked in this manner, all clauses have been assembled. Each clause then receives an extra point, all of which simultaneously undergo a motion perpendicular to each clause gadget, as shown in line j.

▶ **Lemma 25.** *Given an instance S of* MAX-2-SAT *with l variables and m clauses, it is possible to construct the above temporal-sampling of size $n \in O(m^3 r_0/\delta_0)$ in $O(m^3 r_0/\delta_0)$-time.*

Proof. This is immediate by construction. ◀

We are now ready to prove Theorem 5.3.

Proof of Theorem 5.3. Let $\varepsilon, \varepsilon' \in \mathbb{R}$ with $\varepsilon > 0$, $\varepsilon' > 0$ be given. Let $\omega > 6(1/\varepsilon - 1)$. Let S be an instance of MAX-2-SAT with l variables and m clauses which is promised to either admit a solution which satisfies at least $c \cdot m$ clauses (high satisfiability) or does not admit any solution which satisfies more than $s \cdot m$ clauses (low satisfiability), for c and s from Theorem 22. Let P be the result of the above construction with $r_0/\delta_0 = l^\omega$. It follows from Lemma 25 that the size of P is $n \in O(l^{6+\omega})$ since $m \in O(l^2)$. From this and the choice of ω there exists a universal constant c_1 such that $c_1 n^{1-\varepsilon} \leq r_0/\delta_0 = l^\omega$. Suppose there exists some $(c_0, 2 - \varepsilon', c_\delta n^{1-\varepsilon})$-approximation for some $c_0, c_\delta \in \mathbb{R}$ with $c_\delta < c_1$. Run the approximation on P with $k = 2m + (1-c)m$, $r = r_0$, $\delta = \delta_0$. Then either it fails and Lemma 23 implies there does not exist a truth assignment satisfying at least $c \cdot m$ clauses (thereby deciding that it is a low satisfiability instance in polynomial-time), or it outputs a clustering $\mathcal{C}$. Since $c_\delta n^{1-\varepsilon} < r_0/\delta_0$, it is easy to see that Lemma 23 and Lemma 24 still apply. If S is a high satisfiability instance then it follows by Lemma 23 that the resulting clustering consists of at most $c_0 \cdot (2m + (1-c)m)$ clusters with $\mathsf{rad}_\infty(\mathcal{C}) \leq (2 - \varepsilon')r_0$. Otherwise by Lemma 24 it cannot consist of fewer than $2m + (1-s)m$ centers with $\mathsf{rad}_\infty(\mathcal{C}) < 2r_0$. Thus it is possible to distinguish high satisfiability instances from low satisfiability instances provided that $(2 - \varepsilon') < 2$ and $c_0 \cdot (2m + (1-c)m) < 2m + (1-s)m$. The former inequality holds by choice of ε'. From the later inequality it follows that the input classes will be distinguishable provided that $c_0 < (3-s)/(3-c)$. The theorem follows by setting c_0 to be the infimum of the right hand side subject to the constraints placed on c and s in Theorem 22. ◀

E   Inapproximability with exact number of clusters for Temporal k-Median

We remark here that the construction of Appendix B which demonstrates NP-hardness of $(1, \mathsf{poly}(n), \mathsf{poly}(n))$-approximation works for TEMPORAL k-MEDIAN with only minor modification.

Given any positive real numbers ε, $s \leq 1$, r_0, and $\delta_0 < r_0\sqrt{3}/4$, and some l-variable instance S of Exact-3-SAT, using essentially the same construction as used in Appendix B we show how to construct a temporal sampling P such that the following conditions hold:

1. P admits a temporal $(l, \frac{1}{2}r_0(l+1), \delta_0)$-clustering if S is satisfiable.
2. P does not admit a temporal $(l, c_r n^{s(1-\varepsilon)} r_0, c_\delta n^{(1-s)(1-\varepsilon)} \delta_0)$-clustering otherwise.

The only difference will be in our selection of the value of the constant ρ, which we determine later.

▶ **Lemma 26.** *If S is satisfiable then P admits an temporal $(l, \frac{1}{2}r_0(l+1), \delta_0)$-clustering.*

Proof. We make the identical assignments as Lemma 18, the only difference here is that we use the k-median objective. The cost of covering the variable gadgets in *Initial Position*, *Phase 1*, and *Phase 3* are $\frac{1}{2}r_0 l$. In *Phase 2* one of the remaining centers near where the clauses meet needs to cover the points of the variable gadget which gives its center to the extra point. The cost of this level is $\frac{1}{2}r_0(l+1)$. ◀

▶ **Lemma 27.** *If $\mathcal{S}$ is not satisfiable then P does not admit an temporal $(l, \frac{1}{2}r_0 l + \rho r_0, \delta_0)$-clustering.*

Proof. The proof is identical to Lemma 19, except we use the k-median objective and require that $\rho > l + 1$ so that the spacing of the variable gadgets in initial position exceeds the cost of their k-median clustering. ◀

Since the construction is the same as in Appendix B up to the choice of ρ, we can apply Lemma 20 to conclude $|P| \in O(\rho r_0 l^2 m/\delta_0)$.

We are now ready to prove Theorem 7.1.

Proof of Theorem 7.1. Let $c > 5(1/\varepsilon - 1)$. Let $\mathcal{S}$ be an instance of 3-SAT with l variables and m clauses. We invoke the construction of Appendix B with $\rho(l) = l^{s \cdot c}$, $r_0(l)/\delta_0(l) = l^{(1-s) \cdot c}$ and yield, in polynomial-time, a temporal-sampling P over $\mathbb{R}^2$ of size $n \in O(l^{c+5})$. Note here we have used the fact that $m \in O(l^3)$. Suppose the existence of some polynomial-time $(1, \alpha(n), \beta(n))$-approximation where $\alpha(n) = c_r n^{s \cdot (1-\varepsilon)}$, and $\beta(n) = c_\delta n^{(1-s)(1-\varepsilon)}$. There exists a universal constant c_1 (where $c_1 > 1$ by construction) such that $c_1 n^{(1-s) \cdot (1-\varepsilon)} \leq (\sqrt{3}/4)r_0/\delta_0 = (\sqrt{3}/4)l^{(1-s) \cdot c}$. Thus we let $1 \leq c_\delta < c_1$, As with the remark of Appendix B, Lemma 26 and Lemma 27 still apply. We run this approximation on P with $k = l$, $r = \frac{1}{2}r_0(l+1)$, and $\delta = \delta_0$. In polynomial-time the algorithm either produces output or fails. If the algorithm fails it follows by Definition 3 that P does not admit a temporal $(l, \frac{1}{2}r_0(l + 1), \delta_0)$-clustering and thus $\mathcal{S}$ is not satisfiable by Lemma 26. Otherwise, the algorithm produces some clustering $\mathcal{C}$ of radius $\mathsf{rad}_1(\mathcal{C}) \leq \alpha(n)\frac{1}{2}r_0(l + 1)$. Lemma 27 ensures that if $\mathcal{S}$ is not satisfiable then $\mathsf{rad}_1(\mathcal{C}) \geq \frac{1}{2}r_0 l + \rho r_0$. Thus provided that $\alpha(n)\frac{1}{2}r_0(l + 1) \leq \frac{1}{2}r_0 l + \rho r_0$, the algorithm produces output if and only if $\mathcal{S}$ is satisfiable, giving a polynomial-time test for satisfiability. That is, whenever

$$\alpha(n) \leq \frac{l + 2l^{s \cdot c}}{l + 1} < 3l^{s \cdot c} \leq c_r n^{\frac{s \cdot c}{c+5}} \leq c_r n^{s \cdot (1-\varepsilon)},$$

for some constant c_r. ◀

F   NP-**hardness of** $(O(1), O(1), \mathsf{poly}(n))$-**approximation for Temporal** k-**Median**

In this section we show that it is is NP-hard to simultaneously approximate both the number of clusters and the spatial cost arbitrarily well.

▶ **Definition 28.** *A d-regular graph $G = (V, E)$ is an ω-expander if for every set $S \subset V$ where $|S| \leq \frac{1}{2}|V|$ at least $\omega d|S|$ edges connect S and $V \setminus S$.*

Let $\mathcal{S}$ be an instance of MAX-2-SAT consisting of l variables and m clauses. Given $0 < \delta_0 < r_0$, our goal is to construct a temporal-sampling P in polynomial-time such that:

1. P admits a temporal $(2m + (1 - c)m, 7mr_0, \delta_0)$-clustering if there exists a truth assignment that satisfies at least $c \cdot m$ clauses of $\mathcal{S}$.
2. P does not admit a temporal (k, ρ, δ_0)-clustering for any $k < 2m + (1 - s)m - fm$ and $\rho < (7 + c_2 f)mr_0$, for some universal constant $c_2 \geq 0$, and any $0 \leq f \leq 1/2$, if every truth assignment satisfies at most $s \cdot m$ clauses.

Here, c and s are the constants from Theorem 22.
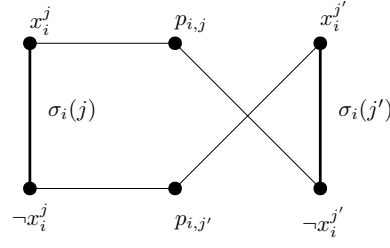
🟨 **Figure 10** All lines indicate a distance of r_0. Dark lines join the points of variable gadgets $\sigma_i(j)$, $\sigma_i(j')$, respectively.

Variable gadgets.

For each variable x_i of $\mathcal{S}$, let k_i be the number of literals where it appears. We introduce k_i pairs of points into each level of P, which we denote by x_i^j and $\neg x_i^j$ for $j \in [k_i]$. We set the distance $d(x_i^j, \neg x_i^j) = r_0$ for all $j \in [k_i]$ in all levels. We think of each pair as a variable gadget for the variable x_i so that any level contains k_i "copies" of this gadget.

Clause gadgets.

We arrange a configuration of points for each clause $c = (l_1 \vee l_2)$ of $\mathcal{S}$ by transporting one of the variable gadget copies of each of its variables to a predetermined location. Ultimately, we overlay them so that the points of each gadget corresponding to l_1 and l_2 overlap on one side, with $\neg l_1, \neg l_2$ overlapping on the other.

> **Phase 1: Initial layout of $P(1)$.** Fix an ω. To determine the initial distances among the variable copies we generate a k_i-vertex 3-regular ω-expander, $G_i = (V_i, E_i)$ for each variable x_i. Fix bijections σ_i from V_i to the set of copies of variable gadgets for x_i in $P(1)$. For each edge $e = (j, j') \in E_i$ we introduce a pair of auxiliary points $p_{i,j}$, $p_{i,j'}$ into $P(1)$ and set the distances between these points and the points of variable gadgets $\sigma_i(j)$, and $\sigma_i(j')$ according to Figure 10. For points of variable gadget copies from distinct variables x_i, $x_{i'}$, we set $d(x_i^j, \neg x_{i'}^{j'}) = d(\neg x_i^j, x_{i'}^{j'}) = (7 + c_2)mr_0$ for some to-be-determined constant, c_2. We think of this process of specifying distances as building a weighted graph on the points of $P(1)$, where the weights are given by the distance values. Thus we allow any remaining unspecified distances of $P(1)$ to be given by the shortest paths distance in this graph.

> **Phase 2: Isolation of variable gadgets.** In $P(1)$ all variable gadget copies are separated by distances of at least $2r_0$. In this phase we simultaneously expand the distances between the points of any pair of variable gadget copies to $(7 + c_2)mr_0$ in steps of size at most δ_0. The only points present in these levels are the $2m$ points of the variable gadget copies. The number of levels required for this stage is $O(mr_0/\delta_0)$

> **Phase 3: Condensation of clause gadgets.** For each clause $(l_i \vee l_{i'})$. Let x_i, $x_{i'}$ denote the variables referenced by literals l_i, and l'_i, respectively. We select $j \in [k_i]$, $j' \in [k_{i'}]$ corresponding to yet unused copies of variable gadgets x_i and x'_i, respectively. By a slight abuse of notation we label the points corresponding to l_i and $l_{i'}$ with the labels l_i, $l_{i'}$. In other words l_i corresponds to $\neg x_i^j$ if it is negated, otherwise x_i^j with $l_{i'}$ defined analogously. We transport each variable gadget so that the points of each which correspond to l_i and $l_{i'}$ overlap on one side, with $\neg l_i, \neg l_{i'}$ overlapping on the other. We call the point where l_i

and $l_{i'}$ overlap the *location* of the clause. The number of levels required for this stage is $O(mr_0/\delta_0)$

Phase 4: Clause verification. Once the clause gadgets have been assembled, we introduce an extra point at the location of each of clause. These extra points move directly away from their respective clauses for a distance of $(7 + c_2)mr_0$. The number of levels required for this stage is $O(mr_0/\delta_0)$

Let P denote the temporal-sampling given by the above construction.

▶ **Lemma 29.** *If there exists a truth assignment satisfying at least $c \cdot m$ clauses, then P admits a temporal $(2m + (1 - c)m, 7mr_0, \delta_0)$-clustering.*

Proof. Let T be a satisfying assignment which satisfies at least c-fraction of the clauses. We now use T to construct a clustering $\mathcal{C}$ with $|\mathcal{C}| \leq 2m + (1 - c)m$, $\delta(\mathcal{C}) \leq \delta_0$. Introduce $2m$ trajectories $\tau_{i,j}$ such that

$$\tau_{i,j}(i') = \begin{cases} x_i^j & \text{if } T(x_i) = \text{TRUE} \\ \neg x_i^j & \text{if } T(x_i) = \text{FALSE} \end{cases},$$

for all $i \in [l]$, for all $j \in [k_i]$, and for all i' starting at the first level of *Phase* 1 through the last level of *Phase* 3. Note that since T satisfies $\mathcal{S}$, every satisfiable clause has at least one trajectory at its *location* at the end of *Phase* 3. We extend one such trajectory to cover the clause's extra point throughout *Phase* 4. The other trajectory which is near to the clause maintains the value that it had at the end of *Phase* 3 through all levels of *Phase* 4. This assigns $2m$ of the at most $2m + (1 - c)m$ trajectories. We will now use the remaining $(1 - c)m$ to pay for the unsatisfiable clauses. For each clause gadget corresponding to some unsatisfied clause, select one of the literals l_i^j at its location (that is, l_i^j is either x_i^j or $\neg x_i^j$ for some $i \in [l]$, $j \in [k_i]$). We introduce a trajectory τ into the clustering such that $\tau = l_i^j$ from the first level of *Phase* 1 through the last level of *Phase* 3, and which covers the clause's extra point throughout each level of *Phase* 4. This completes our assignment. Since there are no more than $(1 - c)m$ unsatisfied clauses, the size of the clustering is at most $2m + (1 - c)m$, as desired. Further, by construction the displacement between trajectory centers on adjacent levels is at most δ_0. Thus $\delta(\mathcal{C}) \leq \delta_0$.

We now argue that $\mathsf{rad}_1(\mathcal{C}) \leq 7mr_0$. Note that in all levels, all points are within a distance of r_0 from some center. In particular, by assignment from T, it holds that for any i either x_i^j has a center for all $j \in [k_i]$ or $\neg x_i^j$ has a center for all $j \in [k_i]$ for all levels of the first three phases. Thus each variable gadget has a center at at least one end, so we can cover all variable gadget points at a cost of mr_0. This same cost of covering the variable gadgets also holds in *Phase* 4, as at least one trajectory remains incident to the overlapping pair of variable gadgets. Since the second through last levels of *Phase* 1, all levels of *Phase* 2, and all levels of *Phase* 3 only contain points which correspond to variable gadgets, these levels be covered at a cost of mr_0. This bound also extends to the points of *Phase* 4 since any extra points which appear in those levels are centers of some trajectory by construction. It only remains to bound the cost of $P(1)$. Recall that there are pairs of extra points in $P(1)$ which correspond to edges of each expander. The fact that all of either the TRUE or FALSE sides of each variable gadget receive a center, implies that all extra points are at a distance of r_0 to a center. Thus the cost of covering the first level is at most $mr_0 + 2r_0 \sum_{i \in [l]} |E_i|$. Since each G_i is 3-regular, $|E_i| \leq \frac{3}{2}|V_i| = \frac{3}{2}k_i$. Thus, $mr_0 + 2r_0 \sum_{i \in [l]} |E_i| \leq mr_0 + 3r_0 \sum_{i \in [l]} k_i = 7mr_0$.   ◀

▶ **Lemma 30.** *If every truth assignment satisfies at most $s \cdot m$ clauses then, for any $0 \leq f < 1$, P does not admit a temporal $(2m + (1 - s)m - fm, (7 + c_2 f)mr_0, \delta_0)$-clustering.*

Proof. Let $0 \leq f < 1$. We will argue that if P admits a temporal (k, ρ, δ_0)-clustering, $\mathcal{C}$, for some $k < 2m + (1-s)m - fm$, and $\rho < (7 + c_2 f)mr_0$, then there exists a truth assignment which satisfies more than $s \cdot m$ clauses. First note that the distance from any extra point of $P(1)$ to its nearest neighbor in $P(2)$ is r_0. Thus, since $\delta_0 < r_0$, no such point has a feasible successor. It follows that none of these points appear in the trajectories of $\mathcal{C}$. Further, there is a unique valid successor for all variable gadget points in all levels of *Phase* 1, *Phase* 2, and all but the later the levels of *Phase* 3 when the corresponding ends of the overlaid variable gadgets come within δ_0 of each other. Since $\mathsf{rad}_1(\mathcal{C}) \leq \rho < (7 + c_2 f)mr_0$, it must be the case that the farthest distance to any cluster center is within $(7 + c_2 f)mr_0$. In particular this means that every variable gadget has been assigned a center. As otherwise some variable gadget at the end of *Phase* 2 is covered by a center of another and the nearest one is at a distance of $(7 + c_2)mr_0$ away. Further, since the extra point of *Phase* 4 also moves a distance of $(7 + c_2)mr_0$, it must also be the case that some trajectory is incident to the location of each clause.

We would now like to infer a truth assignment from $\mathcal{C}$, but the main obstacle is that both x_i^j and $\neg x_i^{j'}$ can be trajectories. We will show that the total discrepancy is bounded, and that even after accounting for "unfairly" satisfied clauses, it is possible to construct a satisfying assignment which satisfies more than $s \cdot m$ clauses. To see this, note that for each pair of trajectories corresponding to the same variable which disagree and share an edge in G_i, the gadget in Figure 10 costs an additional r_0 to cover. For each G_i let I_i denote the subset of vertices of V_i which correspond to the minority assignment. That is, I_i is the subset of vertices of V_i which correspond to the subset of either TRUE or FALSE variable gadgets, whichever has smaller cardinality. It follows that $\frac{1}{m} \sum_{i \in [l]} |I_i|$ is an upper bound on the total fraction of clauses satisfied by inconsistent trajectories. The additional coverage cost of the inconsistencies is equal to the number of edges which cross the $(I_i, V_i \setminus I_i)$-cut, which, since each G_i is a 3-regular ω-expander, is at least $3\omega|I_i|$. Since $\mathsf{rad}_1(\mathcal{C}) \leq (7 + c_2 f)mr_0$, we have that $7mr_0 + \frac{3\omega}{m} \sum_{i \in [l]} |I_i|mr_0 \leq (7 + c_2 f)mr_0$. Thus, $\frac{3\omega}{m} \sum_{i \in [l]} |I_i| \leq c_2 f$. By setting $c_2 = 3\omega$, f is an upper bound on the fraction of clauses flipped by inconsistent trajectories. Let $I : P(1) \to \{\text{TRUE}, \text{FALSE}\}$ such that

$$I(x_i^j) = \begin{cases} \text{TRUE} & \text{if } x_i^j \text{ has a center.} \\ \text{FALSE} & \text{if } \neg x_i^j \text{ has a center.} \end{cases}$$

Consider a truth assignment T such that $T(x_i) = \text{Maj}_{j \in [k_i]}(I(x_i^j))$, and let $|T|$ denote the number of clauses satisfied by T. Since $\mathsf{rad}_1(\mathcal{C}) \leq (7 + c_2 f)mr_0$ it must be the case the extra points for all clauses are covered. That is, T together with $u = k - 2m$ additional trajectories is sufficient for satisfying all clauses, so that $|T| + u \geq m$. Since $k < 2m + (1-s)m - fm$ we have that $u < (1-s)m - fm$. Combining both inequalities we see that $(1-s)m - fm + |T| > |T| + u \geq m$. It immediately follows that $|T| > (s + f)m \geq s \cdot m$. ◄

We are now ready to prove Theorem 7.2.

Proof of Theorem 7.2. Let $\varepsilon > 0$ be given and suppose that there exists a polynomial-time $(\alpha, \beta, c_\delta n^{1-\varepsilon})$-approximation. Let $\mathcal{S}$ be an instance of MAX-2-SAT which is promised to either admit an assignment which satisfies at least c fraction of the clauses, or does not admit an assignment which satisfies more than s fraction. Let $0 \leq f < c - s$. Construct P from $\mathcal{S}$ in $O(m^2 r_0/\delta_0)$ time. Note that we are free to take δ_0 in the construction of P as small as we like, provided that the number of levels of P remains polynomial in the size of $\mathcal{S}$. Using this freedom we take δ_0 such that $c_\delta n^{1-\varepsilon} < r_0/\delta_0$. Run this approximation

on P with $k = 2m + (1 - c)m$, $r = 7mr_0$, $\delta = \delta_0$. If the approximation fails then it must be the case that $\mathcal{S}$ is a No instance. Otherwise, the approximation produces a clustering, $\mathcal{C}$, with at most $\alpha(2m + (1 - c)m)$ trajectories, $\mathsf{rad}_1(\mathcal{C}) \leq \beta 7mr_0$, and $\delta(\mathcal{C}) \leq c_\delta n^{1-\varepsilon}\delta_0$. Note that Lemma 30 and Lemma 29 still hold since $c_\delta n^{1-\varepsilon}\delta_0 < r_0$. Thus, provided that $\alpha(2m + (1 - c)m) < 2m + (1 - s)m - fm$, and $\beta(7mr_0) < (7 + 3\omega f)mr_0$ it is possible to distinguish Yes MAX-2-SAT instances from No instances. That is, for $\alpha < \frac{3-(s+f)}{3-c}$ and $\beta < 1 + \frac{3}{7}\omega f$. Taking $c_r = \frac{3}{7}\omega$ completes the proof. ◄