Point Set Transformations using Given Groups*

Thomas C. van Dijk[1], Erwin Glazenburg[2], Wouter Meulemans[†1], Anna Schenfish[3], and Arjen Simons[†1]

1 **TU Eindhoven**
{t.c.v.dijk|w.meulemans|a.simons1}@tue.nl
2 **Utrecht University**
e.p.glazenburg@uu.nl
3 **KTH Royal Institute of Technology**
schenf@kth.se

Abstract

We study transforming one point set A to an equal-size point set B optimally, where translating any of a predefined set of *groups* of points incurs a cost independent of its size. We consider two objectives: minimizing the number of groups moved (Cardinality) and minimizing the total translation length of the groups (Length). When a matching between A and B is given – that is, we know which point moves to which other point – we prove that minimizing Cardinality is NP-hard in general, but efficiently solvable when the given groups form a hierarchy. Without a prescribed matching, we prove that both minimizing Cardinality and Length is NP-hard.

Lines 174

1 Introduction

Transforming one point set into another via rigid transformations is a widely studied problem with applications in e.g., computer vision, motion analysis, and robotics [2, 6, 7]. Recent work [8] studied how a given matching (bijection) between point sets can be represented using additive translations applied to subsets (*groups*) of points to capture the "visual complexity" of grouped motion; see Figure 1. This framework introduces two dimensions: how to measure the cost of a grouped translation, and constraints on which points can be grouped together. The considered cost measures are *Cardinality* – the number of groups used – and *Length* – the sum of Euclidean norms of translation vectors applied to groups. One of the constraints is the *Given* variant, where the groups that can be used are prescribed – this paper focuses on this setting with given groups.

* Research on the topic of this paper was initiated at the 9th Workshop on Applied Geometric Algorithms (AGA 2025) in Otterlo, The Netherlands.
† W. Meulemans and A. Simons are (partially) supported by the Dutch Research Council (NWO) under project number VI.Vidi.223.137.

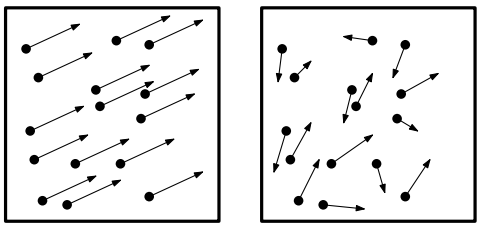

Figure 1 Grouped motion (left) is easier to understand than dispersed motion (right).

42nd European Workshop on Computational Geometry, Hagen, Germany, March 25–27, 2026.
This is an extended abstract of a presentation given at EuroCG'26. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

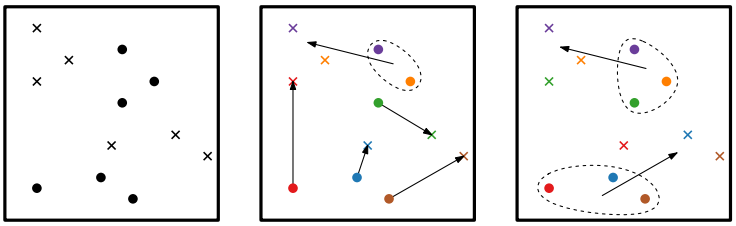

Figure 2 Left: source points A (dots) and target points B (crosses). Middle and Right: different matchings (indicated using colors) result in different grouped motions, with a lower cost on the right.

The above framework [8] considers only settings where a matching between point sets is given. Yet, many applications such as shape matching require determining both the matching and the translations [1, 2, 9]. We thus generalize the framework with another dimension: is the matching given or to be optimized? In the latter, we aim to find the matching minimizing the cost of a grouped translation, subject to constraints on the permitted groups; see Figure 2. This generalization extends the descriptive model of the original framework to a constructive alignment problem related to classical point-set registration and structured motion analysis.

Contributions and organization. Minimizing Cardinality for given groups and matched point sets remained open in [8]: in Section 3 we show that this problem is NP-hard in general, and provide a polynomial-time algorithm when the given groups form a hierarchy. In Section 4 we study the previously unexplored problem where no matching is given, and establish NP-hardness for both optimization criteria. Full proofs can be found in Appendix A.

Related work. For the matched case with given groups, Length can be minimized in (weakly) polynomial time and in linear time when the groups form a hierarchy [8].

Classic registration methods typically assume a single global transformation applied uniformly to all points [9], whereas our setting allows multiple translations tied to predefined groups, enabling motions that cannot be expressed in these classic registration models.

The special case of one group of all points together with a group per point effectively describes classical point-set matching. Minimizing the sum of squared distances is achieved by aligning the centroids [6], while other metrics, like Euclidean distance, are more complicated [4]. Additional work on matching point sets exists [1], differing in the matching distances used, allowed transformations, and whether the point sets have the same cardinality or not. Matching costs are typically based on the L_1 (Manhattan), L_2 (Euclidean), or L_2^2 metrics and take the sum of distances of the matched points, or some bottleneck measure. Such an approach treats points independently, whereas group-based translations couple the motion of points within a group, making such distance-based matchings incompatible with our setting. Other well-known metrics, such as the Earth Mover's distance [3], address weighted point sets, which we do not consider.

2 Preliminaries

Matched. In the *matched* setting of [8], we are given two point sets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ in $\mathbb{R}^d$, with a matching $\phi : A \to B$, such that $\phi(a_i) = b_i$, and a family of subsets $\mathcal{F} \subseteq \mathcal{P}([n])$ of $[n] = \{1, \ldots, n\}$ indicating the groups via point indices. A solution to this problem consists of a pair $(\mathcal{F}', \tau)$, where $\mathcal{F}' \subseteq \mathcal{F}$ and $\tau : \mathcal{F}' \to \mathbb{R}^d$ describes the translation vectors for the groups defined by $\mathcal{F}'$. Let $\mathcal{F}'_i \subseteq \mathcal{F}'$ be only those groups that

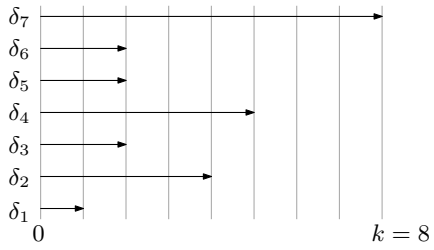

Figure 3 Example construction for $N = \{1, 4, 2, 5, 2, 2\}$, requiring a set N' such that $k = \sum_{x \in N'} x = 8$. Difference vectors $\delta_1, \ldots, \delta_6 \in \Delta$ represent the values in N, and $\delta_7 \in \Delta$ represents k.

contain index i. A solution $(\mathcal{F}', \tau)$ is called *valid* if $\sum_{G \in \mathcal{F}'_i} \tau(G) = b_i - a_i$ for all $i \in [n]$, i.e., if the movement of each group places each point of A to its matched point in B.

For the Cardinality measure, the goal is to minimize $|\mathcal{F}'|$, the number of groups that are translated; for Length, the goal is to minimize the total length translated by all groups, computed as $\sum_{G \in \mathcal{F}'} \|\tau(G)\|$, where $\|\cdot\|$ indicates the Euclidean norm in $\mathbb{R}^d$.

As only the relative positions between matched points are relevant, we can represent the input equivalently by a single collection of vectors $\Delta = \{\delta_1, \ldots \delta_n\}$ with $\delta_i = b_i - a_i$.

Unmatched. We generalize [8] by supposing we must now (also) find the matching $\phi : A \to B$ that minimizes Cardinality or Length. Let Φ be the set of all matchings between A and B. Given two point sets $A, B \subset \mathbb{R}^d$ of equal size n and a family of allowed groups $\mathcal{F}$, we aim to find a matching $\phi^* \in \Phi$ that minimizes Length or Cardinality under the *matched* model described above. Note that each matching ϕ readily defines a collection Δ of difference vectors $\delta_i = \phi(a_i) - a_i$ that can be used as input for the matched problem.

3 Minimizing Matched Cardinality

We prove that, for a given matching, minimizing Cardinality is NP-hard for general $\mathcal{F}$, but polynomial-time solvable if $\mathcal{F}$ is hierarchical.

▶ **Theorem 1.** *Matched Cardinality is NP-hard in $\mathbb{R}^d$ for any $d \geq 1$.*

Proof sketch. For $d = 1$, we use a reduction from Partition, which is weakly NP-hard [5]: given a set $N = \{x_1, \ldots, x_n\}$ of n positive integers, is there a subset $N' \subseteq N$ with $\sum_{x \in N'} x = k$, where $k = \frac{1}{2} \sum_{x \in N} x$? The result for higher d follows trivially.

For each $x_i \in N$, the reduction creates a pair of points in A and B with rightward difference vector δ_i of length x_i; additionally, a single pair with δ_{n+1} of length k to the right (see Figure 3). We set the permitted groups to be each i separately (except $n + 1$), as well as the pairs $\{i, n + 1\}$. If the Partition instance admits a solution, we achieve δ_{n+1} "for free", moving with Cardinality n. Conversely, if the Matched Cardinality instance permits using only n groups, δ_{n+1} must have been achieved for free via some subset of pairs: this subset in N hence sums to k. ◀

Consider the case where the family $\mathcal{F}$ is a hierarchy that includes at least all singleton sets. We now represent $\mathcal{F}$ as a rooted tree T, where leaves correspond to indices in $[n]$, and internal nodes represent the sets induced by their subtree; see Figure 4.

First, consider the restricted case that $\mathcal{F}$ consists only of $[n]$ and all singleton groups. Let $\delta^* \in \Delta$ be the most frequently occurring distance vector. Then δ^* is an optimal translation

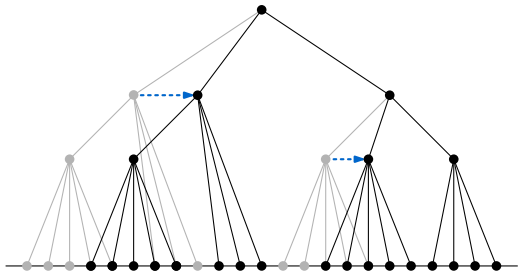

Figure 4 Translating groups in $\mathcal{F}$ (dashed arrows) corresponds to moving entire subtrees.

vector for the root: it aligns the maximum number of leaves; the rest has to be moved to their destination individually. Our algorithm generalizes this idea to arbitrary hierarchical $\mathcal{F}$.

At every node u, we store a set of *preferred vectors* $V(u)$ defined as follows (see also Figure 5). If u is a leaf corresponding to index i, then $V(u) = \{\delta_i\}$. If u is an internal node, then $V(u)$ consists of the most frequent preferred vectors among its children (in case of a tie, it consists of all vectors with maximum frequency).

▶ **Lemma 2.** *In any constant dimension $d \geq 1$, the sets $V(u)$ for all nodes $u \in T$ can be computed in $O(n \log n + nh)$ time, for a tree of height h.*

Proof sketch. We compute $V(u)$ for every node u in a bottom-up fashion. At each node u, we look at the preferred vector sets $V(u_1), \ldots, V(u_k)$ of its children, count how often each vector occurs among those sets, and keep the ones with maximum frequency as $V(u)$. The work at node u is proportional to the sum of sizes of the preferred vector sets of its children.

Any vector in any $V(u)$ is in fact one of the original leaf vectors $\delta_i \in \Delta$, and each δ_i appears at most h times as it can only be part of $V(u)$ for nodes on the path from its leaf to the root. Therefore, a leaf's vector is processed at most h times across the whole tree, giving a time bound of $O(nh)$ if each occurrence can be handled in constant time. This can be achieved using $O(n \log n)$-time preprocessing: sort Δ, arbitrarily but so that duplicate values become adjacent, then assign unique integer identifiers in the range $[n]$ to the distinct δ_i. The required (multi)set operations can be implemented efficiently over this range. ◀

In an eventual solution, all points in the subtree rooted at $u \in T$ are translated according to all translations in the (strict) ancestors of u: we call the sum of these translations the *ancestral translation* of u. Moreover, we use *cost* of u to indicate the minimum number of group translations the subtree requires, given some ancestral translation. Our goal is thus to minimize the cost of the root (which necessarily has an ancestral translation of $\vec{0}$).

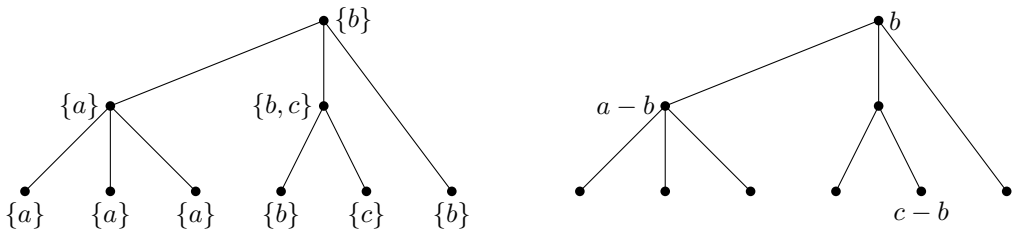

Figure 5 Example two-level hierarchy on six points with translation vectors indicated by letters. (left) Preferred vectors. (right) Minimum Cardinality solution.

▶ **Lemma 3.** *For every node $u \in T$, the cost of u is minimum if its ancestral translation is a preferred vector in $V(u)$. Any other ancestral translation increases this cost by exactly 1.*

Proof. We use induction on the height of the subtree at u. For a leaf node, there is only a single preferred vector in $V(u)$: if the ancestral translation matches this vector, then the subtree is already solved and has cost 0; otherwise, the point needs to move to its target and the subtree has cost 1.

For an internal node, choosing a translation (or none) fixes the ancestral translation for its K children. Any choice can match the preferred vector of a certain number k of children. These k out of the K children receive their minimal cost; the other $K - k$ children have their minimal cost plus 1, by induction. Therefore, the total cost of the children is the sum of their minimum costs plus $K - k$. Clearly, this is minimized by maximizing k, that is, via an ancestral translation that matches a preferred vector in $V(u)$.

If the ancestral translation is not a preferred vector, we can readily apply a single translation at the root of a subtree to make it match a preferred translation: the cost is exactly one more than the minimum cost. ◀

The algorithm now simply computes all preferred vectors in a bottom-up fashion, and then chooses optimal translations in a top-down fashion. We obtain the following result.

▶ **Theorem 4.** *Let Δ be a collection of n vectors in $\mathbb{R}^d$, and let $\mathcal{F}$ be a given hierarchy represented by a tree T with height h. We can compute a solution $(\mathcal{F}', \tau)$ of minimal Cardinality in $O(n \log n + nh)$ time, for any constant dimension $d \geq 1$.*

Proof. By Lemma 2 computing the $V(u)$ vectors for all nodes $u \in T$ takes $O(n \log n + nh)$ time. We recursively compute the translations to apply via a pre-order traversal of T, such that the ancestral translation is known when treating a node. Let node $u \in T$ have an ancestral translation a. If $a \in V(u)$, we apply no translation; otherwise, we select an arbitrary vector $x \in V(u)$ and apply $x - a$. This yields minimal cost by Lemma 3. This recursion runs in $O(nh)$ time, as the total number of preferred vectors on any height is $O(n)$. ◀

4 Computing an optimal matching

We prove that, given two point sets without a matching and a family of allowed groups, computing an optimal matching is NP-hard for both the Cardinality and Length measures.

▶ **Theorem 5.** *Unmatched Cardinality is NP-hard in $\mathbb{R}^d$ for any $d \geq 1$.*

Proof sketch. For $d = 1$, we reduce from 3-Partition, which is strongly NP-hard [5]: given a set $N = \{x_1, \ldots, x_{3n}\}$ of $3n$ positive integers with $t = \frac{1}{n}\sum_{i=1}^{3n} x_i$, such that all $t/4 < x_i < t/2$, can N be partitioned into triples such that each triple sums to t? The result for higher d follows trivially.

For each $i \in [3n]$, we define a set A_i of x_i points, with unit distance spacing; each A_i is placed at distance 2 from A_{i-1}. See Figure 6. We use $A = \bigcup_i A_i$ and family $\mathcal{F}$ of permitted groups consists of all A_i, as well as all singletons. For each $i \in [n]$, we define a set B_i of t points, with unit distance spacing; each B_i is placed at distance 2 from B_{i-1}. We use $B = \bigcup_i B_i$, and ensure they are placed disjoint from all points in A.

If the 3-Partition instance admits a solution, then each triple $\{x_i, x_j, x_k\}$ in the solution is matched to a unique set B_x and we can move the sets A_i, A_j, A_k to match exactly to the points in B_x. This uses $3n$ groups in total.

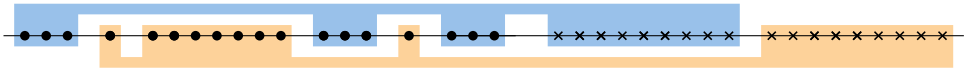

Figure 6 Example construction for Theorem 5 using the set $N = \{3, 1, 7, 3, 1, 3\}$. The points of A (dots) are grouped by these values. We place points of B (crosses) into groups of size $t = 9$. A matching minimizing Cardinality is indicated by the colored regions, and corresponds to a 3-partition.

A solution to the Unmatched Cardinality instance with $3n$ groups must move exactly all A_i groups, as each point in A must move. Each group A_i must move to a single B_x group, due to the spacing between the groups in B, and each point in A maps to a unique point in B. As $t/4 < x_i < t/2$, this implies that each B_x is covered by exactly three A_i sets. ◀

▶ **Theorem 6.** *Unmatched Length is NP-hard in $\mathbb{R}^d$ for any $d \geq 1$.*

Proof sketch. We adapt the reduction of Theorem 5 by introducing sufficient spacing between A and B, such that using an additional group is too costly in terms of Length. ◀

5 Future work

The framework of [8] considered several variants of the matched problem in which the groups are not given as part of the input, under different permitted-groups constraints. We have additional preliminary results indicating that most unmatched variants are NP-hard, making the study of approximability or parameterized complexity a natural direction of future work.

References

1 Helmut Alt and Leonidas J Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In *Handbook of computational geometry*, pages 121–153. Elsevier, 2000.

2 K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, 1987.

3 Scott Cohen and L Guibas. The Earth Mover's distance under transformation sets. In *Proc. 7th IEEE International Conference on Computer Vision*, volume 2, pages 1076–1083. IEEE, 1999.

4 David Eppstein, Marc van Kreveld, Bettina Speckmann, and Frank Staals. Improved grid map layout by point set matching. *International Journal of Computational Geometry & Applications*, 25(02):101–122, 2015.

5 Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1990.

6 Berthold Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society A*, 4:629–642, 1987.

7 Liang Li, Ming Yang, and Chunxiang Wang. Graph correspondence-based point set registration. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 54(7):4101–4112, 2024.

8 Wouter Meulemans, Arjen Simons, and Kevin Verbeek. Visual Complexity of Point Set Mappings. In *SOFSEM 2025: Theory and Practice of Computer Science*, pages 157–171. Springer Nature Switzerland, 2025.

9 Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2262–2275, 2010.

A Full proofs

▶ **Theorem 1.** *Matched Cardinality is NP-hard in $\mathbb{R}^d$ for any $d \geq 1$.*

Proof. For $d = 1$, we use a reduction from Partition, which is weakly NP-hard [5]: given a set $N = \{x_1, \ldots, x_n\}$ of n positive integers and $k = \frac{1}{2}\sum_{x \in N} x$, is there a subset $N' \subseteq N$ with $\sum_{x \in N'} x = k$? The result for higher dimensions follows trivially from this reduction.

Given a Partition instance N, we construct a Matched Cardinality instance: a set Δ of vectors and a family $\mathcal{F}$ of groups (see Figure 3). For each integer $x_i \in N$ we add a vector $\delta_i = x_i$ to Δ. Furthermore, we add the vector $\delta_{n+1} = k$. Finally, we construct $\mathcal{F}$ as follows: for each $x_i \in N$, we add both the singleton $\{i\}$ and the pair $\{i, n+1\}$.

We argue that a valid solution $(\mathcal{F}^*, \tau^*)$ with cardinality $|\mathcal{F}^*| \leq n$ exists if and only if N admits a partition N^*, such that $\sum_{x \in N^*} x = \sum_{x \in N \setminus N^*} x$.

Let N^* be a solution to the Partition instance. For each $x_i \in N^*$ we translate pair $G_i = \{i, n+1\} \in \mathcal{F}$ by $\tau(G_i) = x_i$; for each $x_i \in N \setminus N^*$ we translate singleton $G_i = \{i\} \in \mathcal{F}$ with $\tau(G_i) = x_i$. Each point a_i for $i \in [n]$ is moved by $x_i = \delta_i$, and point a_{n+1} is moved by $\sum_{N^*} x_i = k$: this is a valid solution using n groups.

Let $(\mathcal{F}^*, \tau^*)$ be a valid solution to the Matched Cardinality instance with $|\mathcal{F}^*| \leq n$. Since each group in $\mathcal{F}$ contains at most one index from $[n]$, covering all n indices requires at least n groups. Hence, $|\mathcal{F}^*| = n$ and each $i \in [n]$ must be covered by either its singleton $\{i\}$ or its pair $\{i, n+1\}$. Let I be the set of indices in $[n]$ of points that moved using its pair. Since $\delta_{n+1} = k$, it follows that $\sum_{\delta_i \in I} \delta_i = k$. Thus, the set $N' = \{x_i \mid i \in I\}$ sums to k and is a solution to the Partition instance. ◀

▶ **Lemma 2.** *In any constant dimension $d \geq 1$, the sets $V(u)$ for all nodes $u \in T$ can be computed in $O(n \log n + nh)$ time, for a tree of height h.*

Proof details. The remaining challenge from the main text is as follows: given a tree with an integer label $V(u) \in [n]$ on each leaf u, annotate each internal node u with the set $V(u)$ of labels that occur most frequently in the $V(u)$ sets of its children (which could be more than one label in case of a tie for the maximum). Below, we show how to perform this annotation in $O(nh)$ time, where n is the number of leaves and h the height of the tree.

We represent the sets $V(u)$ as linked lists and construct them in a postorder traversal of the tree. Leaves are a special case: their label is given. For internal nodes, we do the following, after once allocating an array of n integers and initializing it to all zeroes; call it the *accumulator* array.

For an internal node u, we can easily iterate over the labels of its children in time proportional to the number of labels. First, we perform such an iteration, incrementing the element of the accumulator array by one for each label. During this iteration, we maintain the maximum value of the accumulator array. Next, we perform another iteration over the labels[1]. If the label's value in the accumulator array matches the maximum value, we add it to $V(u)$, and in any case we set the element of the accumulator array to zero afterwards. This ensures that each maximum label is added exactly once to $V(u)$ and that the accumulator array resets to all zeroes after processing the node u and can therefore be reused.

In this manner, processing a single label of a child takes constant time and processing an internal node thus takes time proportional to the total number of labels of its children. ◀

[1] The goal is to efficiently iterate over the nonzero elements of the accumulator array: note that there is not enough time to simply scan and reset the entire array at every node.

▶ **Theorem 5.** *Unmatched Cardinality is NP-hard in $\mathbb{R}^d$ for any $d \geq 1$.*

Proof. For $d = 1$, we use a reduction from 3-Partition, which is strongly NP-hard [5]: given a set $N = \{x_1, \ldots, x_{3n}\}$ of $3n$ positive integers and $t = \frac{1}{n}\sum_{i=1}^{3n} x_i$, can N be partitioned into triples such that each triple sums to t? We assume that $t/4 < x_i < t/2$ for all i, such that a subset can achieve a sum of t only if it is indeed a triple [5]. Furthermore, we assume all $x_i > 1$, which can easily be achieved by adding 1 to all values if there are $x_i = 1$ (and increasing t by 3) as this preserves the validity of any triple. The result for higher dimensions follows trivially from this reduction.

We construct an Unmatched Cardinality instance as follows. Each $i \in [3n]$, we define a set A_i of x_i points, with unit distance spacing; each A_i is placed at distance 2 from A_{i-1}. We use $A = \bigcup_i A_i$ and family $\mathcal{F}$ of permitted groups consists of all A_i, as well as all singletons. For each $i \in [n]$, we define a set B_i of t points, with unit distance spacing; each B_i is placed at distance 2 from B_{i-1}. We use $B = \bigcup_i B_i$, and ensure they are placed disjoint from all points in A.

We argue that a valid solution to the Unmatched Cardinality instance with cardinality at most $3n$ exists if and only if the 3-Partition instance has a solution.

Assume there is a solution to the 3-Partition instance. Consider each triple $\{x_i, x_j, x_k\}$ in this solution, and match it to a unique set B_x. We move the sets A_i, A_j, A_k via a translation each to match exactly to the points in B_x – constructing matching ϕ accordingly. Each of the n triples is matched to one of the n sets B_x. As such, we constructed a solution that uses exactly $3n$ group translations.

Assume there is a solution to the Unmatched Cardinality instance of cardinality at most $3n$. As each point in A must move, we must use at least $3n$ groups and we can only use this few groups if we select precisely all A_i groups, and no singletons. As there is a distance of two between two B_x groups, each group A_i must be moved to lie wholly within one of the B_x groups. Since the matching ϕ must match each point from A to a unique point in B, we know that each B_x set must be covered by exactly three A_i sets. This readily defines a 3-Partition solution. ◀

▶ **Theorem 6.** *Unmatched Length is NP-hard in $\mathbb{R}^d$ for any $d \geq 1$.*

Proof. We use effectively the same construction as in the proof of Theorem 5. However, we are more careful about the distance between the A and B sets, such that using more groups is always more costly in terms of total length than using fewer groups.

Erwin: new? Specifically, let $L_A = 2 \cdot (3n-1) + \sum_i (x_i - 1) = 3n - 2 + \sum_i x_i$ be the total distance between the leftmost and rightmost point in A; similarly, $L_B = t - 1$ denotes the distance between the leftmost and rightmost point of one set B_i. Let $D = 3n(L_A + (n+1)L_B/2) + 1$. We place B_1 at distance D from A, and place each B_i at distance D from B_{i-1}.

We argue that a valid solution to Unmatched Length with total length less than $(3n(n+1)/2+1)D$ exists if and only if the 3-partition instance has a solution.

For a 3-Partition solution, we use the same matching argument as before. We have three groups from A that move to each B_i, each of which with length between $iD + (i-1)L_B$ and $iD + iL_B + L_A$; this "slack" of $L_A + L_B$ is introduced because it is unknown from where in A exactly the group moves, and to where in B exactly. The total length of all movements is thus at most $\sum_{i=1}^{n} 3(iD + iL_B + L_A) = 3(\sum_{i=1}^{n} i(D + L_B) + \sum_{i=1}^{n} L_A) = 3((D+L_B)n(n+1)/2 + nL_A) = (3n(n+1)/2)D + 3n((n+1)/2L_B + L_A) = (3n(n+1)/2+1)D - 1$, which is indeed less than $(3n(n+1)/2+1)D$.

For an Unmatched Length solution of length less than $(3n(n+1)/2+1)D$, we argue as follows. By the restriction that $t/4 < x_i < t/2$ for all i we know that at least three groups must move to each set B_i

Erwin: old

Specifically, let $L_A = 2 \cdot (3n-1) + \sum_i (x_i - 1) = 3n - 2 + \sum_i x_i$ be the total distance between the leftmost and rightmost point in A; similarly, $L_B = n(t-1) + n - 1 = nt - 1$ denotes the total distance between the leftmost and rightmost point in B. We place B to the right of A, such that its leftmost point has distance $D = 3n(L_A + L_B) + 1$ to the rightmost of A. Now, any translation that directly moves a point from A to a point from B has length at most $L = L_A + L_B + D$, and at least D.

For a 3-Partition solution, we use the same argument as before. With exactly $3n$ groups moving, we hence readily have that the total length is at most $3nL$.

For a Unmatched Length solution of length at most $3nL$, we argue as follows. A solution that uses $3n+1$ groups uses at least distance $(3n+1)D$. However, $(3n+1)D \leq 3nL$ simplifies to $D \leq 3n(L_A + L_B)$, which is lower than the assigned value for D. As such, the solution must use $3n$ groups: from here, we use the same argument as before. ◀