A COMPLETE COMPLEXITY DICHOTOMY FOR DECIDING CLUSTER IN-BETWEENNESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Unsupervised learning reads every object through a binary lens: inlier or outlier. Recent work adds a third primitive, the *in-between instance* (IBI): an object atypical for every cluster yet simultaneously compatible with several of them, and whose interest lies in *which* traits it inherits from *which* neighbouring group. Existing formulations answer this with a proximity predicate and a scalar score. We instead pose in-betweenness as an exact reconstruction question: can a bounded number of *actual* cluster members be selected so that, coordinate by coordinate, they reproduce a target? The selection is then itself the attribution. We formalize this as INBETWEEN-DECISION, a group-budgeted covering problem over symbolic data, and settle its complexity along five structural parameters: alphabet size σ, dimensionality δ, number of clusters ν, cluster size ξ, target size τ. We obtain a complete dichotomy over all $2^5 = 32$ bounding regimes: solvable in polynomial time exactly when $\{\sigma, \delta\}$, $\{\nu, \xi\}$ or $\{\delta, \tau\}$ is bounded, each witnessed by a fixed-parameter algorithm and NP-complete otherwise. In-between detection is thus feasible precisely when one of three structural budgets is paid.

1 INTRODUCTION

Breaking the dichotomy in machine learning. Clustering, which groups similar objects and separates dissimilar ones, and outlier detection which singles out objects deviating substantially from the data, together induce a binary perspective on every object: it is either an inlier or an anomaly. Kazempour et al. (2024) argue that this dichotomy is incomplete: a *third*, qualitatively distinct kind of object that is neither a cluster member nor an anomaly, but a *connector* exhibiting traits drawn from two or more groups, an *in-between instance* (IBI). An outlier is far from every cluster and close to none, whereas an IBI is located in neighborhood of at least two clusters.

Why IBIs? On the need for a third machine learning primitive. The value of an IBI lies not in flagging an anomaly but in identifying objects that link otherwise separate groups, and in saying *which* groups it links e.g. topic-bridging publications or compounds shared between two pharmacological classes (Kazempour et al. (2024)). IBIs matter especially where temporal, spatial or state transitions are under investigation, such as land-to-water transitions in remote sensing (Sch et al. (2026)), or their application debut on isotope signatures in archeometry (Göhring et al. (2024; 2025)). While these instantiations are continuous, we address the discrete counterpart, where objects are aligned symbolic feature vectors, e.g. categorical records, or discretised measurements.

From a perceptual criterion to a combinatorial question. The vision of Kazempour et al. (2024) is predicate-driven: o is an IBI if a proximity predicate $\theta(o, C_i, C_j)$, e.g. neighbourhood-, probability-, or geometry-based holds for two distinct clusters. This answers "*is this particular object in-between?*". We ask at that point the following question:

> *Given clusters described by the feature values their members realise, can a bounded number of representatives be selected from each cluster such that, coordinate by coordinate, they exactly reconstruct a target object (or set of objects)?*

This replaces a geometric score by the *composability* of a candidate out of its surrounding clusters: the object is a coordinate-wise mosaic of a few actual cluster members. The witness is then *itself* the explanation, meaning that it states, per coordinate, which cluster supplied which symbol, with real

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

objects rather than fitted weights and *in-betweenness turns from a scoring problem into a budgeted covering problem*, open to combinatorial analysis. The reading is natural whenever objects are aligned feature vectors over a discrete alphabet e.g. genomic k-mers, categorical records, tokenised texts where sharing a trait means agreeing on the symbol at a position. We develop the formalism alongside DNA-style instances in the upcoming sections. Being exact and symbol-level rather than metric, it complements rather than replaces continuous instantiations of in-betweenness.

Contributions. What makes in-betweenness hard is not clear *a priori*: the problem couples five structural quantities alphabet size σ, dimensionality δ, number of clusters ν, cluster size ξ, target size τ each small in some application regimes and large in others.

- We formalise in-betweenness as a coordinate-wise covering relation (a *vector element-cover*, VE-cover) between selected cluster representatives and a target, and pose the associated decision problem INBETWEEN-DECISION, with worked DNA-style yes- and no-instances
- We prove NP-completeness for four maximal parameter regimes by reductions from 3-CNF-SAT, and give polynomial-time algorithms for three minimal regimes, each fixed-parameter tractable and, for $\{\sigma, \delta\}$ and $\{\delta, \tau\}$, fixed-parameter *linear* in the input size.
- Together these yield a complete dichotomy over all $2^5 = 32$ bounding regimes: polynomial-time solvable whenever at least one of $\{\sigma, \delta\}$, $\{\nu, \xi\}$, $\{\delta, \tau\}$ is bounded, NP-complete otherwise. The dichotomy doubles as a design rule: it tells a practitioner exactly which structural budget must be paid for in-between detection to be feasible at all.

Scope. This work is theoretical in nature. We introduce a formal decision problem for exact symbolic in-betweenness and characterize its computational complexity over all combinations of five structural parameters. Our main claims are worst-case statements established by proofs, together with fixed-parameter algorithms for the tractable cases, rather than empirical claims about performance on a particular collection of datasets. We therefore do not include an experimental evaluation of this work.

We accompany the formal development with examples phrased as DNA-style sequences, so that the abstract covering relation can be read directly as positional agreement between nucleotide strings:

Example 1 (A DNA yes-instance)**.** *Let* $\Sigma = \{A, C, G, T\}$*,* $d = 4$*, and*

$$A_1 = \{ACGT, ATGT, ACAT\}, \qquad A_2 = \{GCGA, GGTA, TCGC\},$$

with budgets $k_{A_1} = k_{A_2} = 2$ *and target* $I = \{ACGA, ATGT\}$*. Selecting* $A_1' = \{ATGT\}$ *and* $A_2' = \{GCGA\}$ *VE-covers* I*: the column sets of the selection are* $\Pi_1 = \{A, G\}$*,* $\Pi_2 = \{T, C\}$*,* $\Pi_3 = \{G\}$*,* $\Pi_4 = \{T, A\}$*, and these contain, column by column, every symbol required by* $ACGA$ *and* $ATGT$*. Hence the instance is a* yes*-instance. Figure 1(a) shows the selection; the witness is minimal in that it uses one representative from* each *family, exhibiting* I *as a genuine two-cluster recombination.*

Example 2 (A DNA no-instance)**.** *Over the same alphabet with* $d = 4$*, let* $A_1 = \{ACGT, ATGT\}$*,* $A_2 = \{GCGA, GGTA\}$*, budgets* $k_{A_1} = k_{A_2} = 2$*, and target* $I = \{ACCA\}$*. No selection can VE-cover* I*: the third column of* I *demands the symbol* C*, but no available sequence has* C *in its third position, so* $\Pi_3(\textit{selection})$ *can never contain* C *(Figure 1(b)). The instance is therefore a* no*-instance, regardless of budgets. This isolates the mechanism by which the problem fails: a single column whose required symbol is unavailable suffices to rule out in-betweenness.*

In example 1 the target is not equal to any cluster member, yet each of its feature values is realized by a nearby cluster, so I is reconstructible from a handful of representatives. We stress that VE-covering is an *exact, symbol-level* relation, not a metric one; it is the natural primitive when features are categorical and "proximity" means "agreement on a coordinate". Metric and probabilistic instantiations of in-betweenness (neighborhood, class-probability, and angle criteria) remain available in continuous settings and are complementary to the combinatorial core studied here.

2 RELATED WORK

While Kazempour et al. (2024) established the in-between instance (IBI) as a first-class primitive using *continuous* proximity predicates, in this work, we harden the IBI concept for discrete representation learning by proposing exact symbol-level reconstruction as the rigorous categorical analogue of

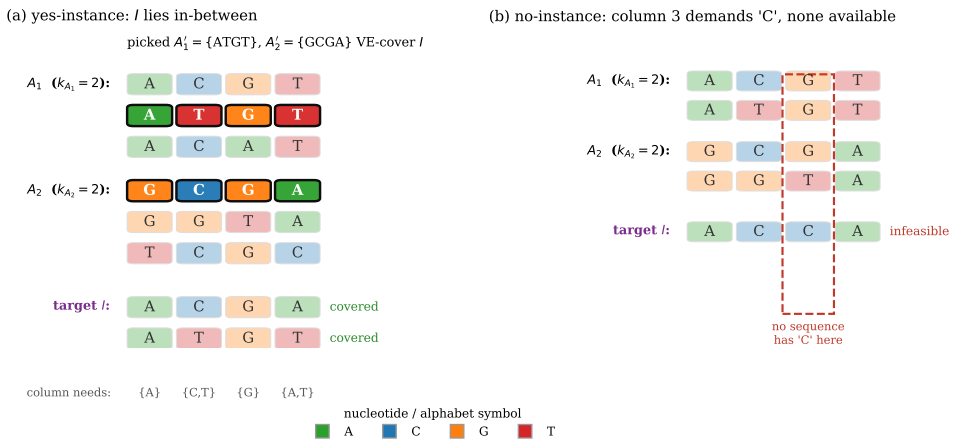

Figure 1: The covering relation as positional agreement between DNA-style sequences. **(a)** The yes-instance of Example 1: the picked sequences ATGT$\in A_1$ and GCGA$\in A_2$ (bold) jointly supply, in every column, all nucleotides required by the targets ACGA and ATGT; the per-column requirement is shown beneath. **(b)** The no-instance of Example 2: column 3 of the target requires C, which no sequence provides, so the instance has no VE-cover for any budget.

metric in-betweenness. INBETWEEN-DECISION is positioned against four archetypes within the literature (Table 1).

Soft, Subspace, and Density-Based Clustering. Fuzzy c-means (Bezdek (1981)) and Gaussian mixtures (McLachlan & Peel (2000)) assign graded memberships or posterior responsibilities, but output a *single scalar summary per object-cluster pair*. This, fails to indicate *which features* are inherited from *which cluster*, rendering IBIs indistinguishable from tail-end outliers. Subspace clustering and biclustering identify clusters within localized feature subsets, yet they group multiple objects into a shared subspace rather than attributing the disjoint feature subsets of a *single* object to multiple distinct clusters simultaneously. Similarly, density-based methods such as DBSCAN (Ester et al. (1996)) can identify low-density corridors (border points), but treat them as artifacts rather than explicitly reconstructing their features from flanking dense regions.

Anomaly Detection & Continuous Embeddings. Standard anomaly detection (Ruff et al. (2021)) collapses deviations into a single scalar score, fundamentally conflating outliers (that are far away from any clusters) with IBIs (that are bridging multiple clusters). While modern foundation models and contrastive learning can project discrete symbols into continuous embedding spaces, where geometric anomaly metrics apply, this projection inherently destroys exact symbolic provenance. We argue in this work that for discrete ML paradigms (e.g. genomics, categorical tabular data etc.), explainable attribution must be solved in the native symbolic space to preserve exactness.

Source Attribution & Prototypical Explainability. Models like Latent Dirichlet Allocation (Blei et al. (2003)) provide coordinate-level provenance via probabilistic mixtures over *latent, fitted* sources. In contrast, our formulation investigates exact reconstruction from *given* clusters using a bounded number of actual data representatives. Recent deep architectures like Concept Bottleneck Models (Koh et al. (2020)) and Prototypical Part Networks (ProtoPNets) (Chen et al. (2018)) offer budgeted, exact witnesses tied to specific coordinates. However, these methods explain *supervised* class decisions using *intra-class* prototypes. They do not solve the problem of explaining an anomalous object as an *inter-class* composite of neighboring clusters.

The Combinatorial Limits of Explainability. By expressing in-betweenness as a structural question, namely "Can a target be exactly reconstructed by a budgeted selection of neighbors?", we shift the problem to a group-budgeted set cover (Karp (1972)), related to closest string selection (Gramm et al. (2003)). Rather than a mere algorithmic reduction, our resulting dichotomy determines under which structural conditions this form of exact, budgeted attribution is tractable. It addresses the question under what structural conditions an exact coordinate-wise feature attribution is computationally tractable?

Table 1: Positioning of INBETWEEN-DECISION. *Between-space*: models the region between clusters? *IBI ≠ outlier*: separates a connector from an anomaly? *Coord. attr.*: maps specific coordinates to specific groups? *Exact witness*: uses real data objects for exact reconstruction? *Unsupervised*: operates without class labels? *Budgeted*: bounds the number of contributing sources? The criteria reflect the goals of this work, not the overall quality.

Paradigm	Between-space	IBI ≠ outlier	Coord. attr.	Exact witness	Unsupervised	Budgeted
Soft/GMM (Bezdek, 1981; McLachlan & Peel, 2000)	partly	partly	no	no	yes	no
Subspace Clustering / Biclustering	no	no	partly	no	yes	no
Anomaly detection (Ruff et al., 2021)	no	no	no	no	yes	no
Topic Models / LDA (Blei et al., 2003)	yes	partly	yes	no	yes	no
Feature attribution (Lundberg & Lee, 2017)	no	no	yes	no	partly	no
ProtoPNets / CBMs (Chen et al., 2018)	no	no	yes	yes	no	yes
Continuous IBI (Kazempour et al., 2024)	yes	yes	no	no	yes	no
INBETWEEN-DECISION (this work)	**yes**	**yes**	**yes**	**yes**	**yes**	**yes**

3 FORMAL INVESTIGATION OF THE INBETWEEN-DECISION PROBLEM

We now make the question from Section 1 precise and formally investigate the problem of deciding whether a target object, or a small set of target objects (see Remark 17 in Appendix A.1), can be reconstructed feature by feature from a budgeted selection of actual cluster members.

We begin by introducing the necessary notation and formal framework. Let Σ be an alphabet of distinct comparable elements (e.g., $\Sigma \subseteq \mathbb{N}$ or $\Sigma \subseteq \mathbb{Q}$). For $m \in \mathbb{N}$, define $[m] = \{1, \ldots, m\}$. Let $d \in \mathbb{N}$. A d-dimensional data point is a vector $p \in \Sigma^d$, with $p[i]$ denoting its ith component, and a d-cluster is a set $A \subseteq \Sigma^d$. Given two d-clusters A and B, we say that A *vector-element-covers* (*VE-covers*) B if, for every $p \in B$ and every $i \in [d]$, there exists some $v \in A$ with $p[i] = v[i]$. Thus, the coordinates of a single vector in B may be covered by different vectors of A. This suffices to define the following decision problem:

[`Inbetween-Decision`] Given an alphabet Σ, numbers $d, n \in \mathbb{N}$, d-clusters A_1 to A_n, numbers k_{A_1},...,$k_{A_n} \in \mathbb{N}$, and a d-cluster I, do there exist, for each $i \in [n]$, subsets $A'_i \subseteq A_i$ with $|A'_i| \leq k_{A_i}$ such that $\bigcup_{i=1}^{n} A'_i$ VE-covers I?

Example 18 in Appendix A.1 provides an illustration of this problem. We observe five problem parameters: alphabet size $|\Sigma|$, dimensionality d, number of input clusters n, maximum input-cluster size $\max_{i \in [n]} |A_i|$, and target-cluster size $|I|$. We denote these five parameters by σ, δ, ν, ξ, and τ, respectively. Using this notation, we call a set $P \subseteq \{\sigma, \delta, \nu, \xi, \tau\}$ a *parameter-prefix* (or prefix). By P-`Inbetween-Decision`, we denote the corresponding problem variant in which the parameters in P are bounded by constants, while all remaining parameters are left unbounded. For example, $\{\sigma, \delta, \xi\}$-`Inbetween-Decision` bounds the alphabet size, dimensionality, and maximum input-cluster size, but leaves the number of clusters and the target-cluster size unbounded.

The unrestricted problem corresponds to the empty prefix $P = \emptyset$, whereas $P = \{\sigma, \delta, \nu, \xi, \tau\}$ bounds all five parameters. In the latter case, exhaustive enumeration of the possible selections requires polynomial time. The central question is therefore which intermediate prefixes already suffice to make the problem (in)tractable. The classification is monotone with respect to these prefixes. NP-hardness for some prefix P implies NP-hardness for every prefix $P' \subseteq P$, while polynomial-time decidability for P implies polynomial-time decidability for every $P' \supseteq P$. Indeed, removing a bounded parameter only generalizes the problem, whereas adding one only restricts its instances. Hence, it suffices to identify the boundary cases rather than consider all parameter-prefixes individually.

In the following, we will obtain a complete dichotomy by showing that the problem is polynomial-time decidable whenever the prefix contains at least one of $\{\sigma, \delta\}$, $\{\nu, \xi\}$, or $\{\delta, \tau\}$, and is NP-complete otherwise. The complete classification is summarized in Figure 2.

Before separating NP-hard and polynomial-time cases, we observe that all variants are in NP.

Proposition 3. *Let $P \subseteq \{\sigma, \delta, \nu, \xi, \tau\}$. Then, P-`Inbetween-Decision` is in NP.*

Indeed, a certificate consists of the selected subsets $A'_i \subseteq A_i$. We can verify the bounds $|A'_i| \leq k_{A_i}$ and then, for every $p \in I$ and every coordinate $\ell \in [d]$, check whether some selected vector $v \in \bigcup_{i=1}^{n} A'_i$ satisfies $v[\ell] = p[\ell]$. Both steps are polynomial in the input size. The full proof is given in Appendix A.1.1.

3.1 NP-Hardness Boundary

Together with Proposition 3, the following four results establish NP-completeness for all maximal hard parameter regimes. All reductions are from the NP-complete problem 3-CNF-SAT. Let

$$\varphi = C_1 \wedge \cdots \wedge C_n$$

be a 3-CNF formula over variables $X_1, \ldots, X_m$, where every clause consists of three literals. Full proofs and extended examples are given in Appendix A (with detailed subsections mentioned later).

The four reductions can be read as two pairs. The first two keep the alphabet binary and encode the formula through an increasing dimensionality. The latter two instead fix the dimensionality to three and move the information into the alphabet. Within each pair, the first construction uses a single large input cluster, while the second distributes the choices over many clusters of constant size. This exposes the different structural properties through which hardness can arise. In particular,

Remark 4. *Note that, even though two of the provided following reductions technically only use one input cluster, all constructions can be naturally extended to a* true *IBI setting of two or more input clusters by adding arbitrary clusters that are irrelevant for the targets of the constructed input instance. Hence, hardness is preserved when multiple clusters are allowed.*

Proposition 5. *Let $P \subseteq \{\sigma, \nu, \tau\}$. Then, P-*`Inbetween-Decision` *is NP-complete.*

For the reduction, set $\Sigma = \{0, 1\}$, use one input cluster A_1, and let $d = m + n$. The target I consists only of the all-one vector. For every variable X_i, we add two literal vectors to A_1, one for X_i and one for $\neg X_i$. Both contain 1 in coordinate i and 0 in all other variable coordinates. In clause coordinate $m + j$, a literal vector contains 1 exactly if its literal occurs in C_j. Finally, set $k_{A_1} = m$.

The construction therefore separates the coordinates into two roles. The first m coordinates encode the choice of a truth value for every variable, while the remaining n coordinates test whether the resulting assignment satisfies the clauses. Indeed, the first m target coordinates force the selection of one literal vector for every variable and the bound permits no additional vectors. The remaining coordinates are covered exactly when every clause contains a selected literal. Thus, I is VE-covered if and only if φ is satisfiable. See Appendix A.2.1 for the full proof and an extended example.

Proposition 6. *Let $P \subseteq \{\sigma, \xi, \tau\}$. Then, P-*`Inbetween-Decision` *is NP-complete.*

Here, set again $\Sigma = \{0, 1\}$, but let $d = n$. For every variable X_i, construct a cluster

$$A_i = \{p_i, \overline{p}_i\}$$

with $k_{A_i} = 1$, where $p_i[j] = 1$ exactly if X_i occurs in C_j, and $\overline{p}_i[j] = 1$ exactly if $\neg X_i$ occurs in C_j. The target again consists only of the all-one vector.

A satisfying assignment provides a VE-cover by selecting the vector corresponding to the true literal of every variable. Conversely, any VE-cover selects at most one polarity per variable, and every clause coordinate must be covered by some selected literal vector. The resulting partial truth assignment satisfies every clause and can be extended arbitrarily on unselected variables. Hence, the constructed instance is a yes-instance exactly when φ is satisfiable. The full proof and an extended example are given in Appendix A.2.2. Here, consider the following example sketch.

Example 7. *For*

$$(X_1 \vee X_2 \vee \neg X_3) \wedge (\neg X_1 \vee X_2 \vee X_4)$$

the construction gives

$$I = \left\{\begin{pmatrix}1\\1\end{pmatrix}\right\} \quad A_1 = \left\{\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}\right\} \quad A_2 = \left\{\begin{pmatrix}1\\1\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}\right\} \quad A_3 = \left\{\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right\} \quad A_4 = \left\{\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}\right\}.$$

Each coordinate now represents one clause. For example, selecting $(1, 1)$ from A_2 already covers the target and corresponds to the partial assignment $X_2 = 1$, which satisfies both clauses. This illustrates why the reverse direction only needs to recover a partial satisfying assignment.

Proposition 8. *Let $P \subseteq \{\delta, \nu\}$. Then, P-*`Inbetween-Decision` *is NP-complete.*

We now keep the dimensionality constant and encode the required information in the alphabet. Set $d = 3$ and use a single input cluster. Introduce distinct symbols $x_{i,j}$ for $i \in [m]$ and $j \in [n]$, one symbol c_j for every clause, and an additional symbol $\#$. For each variable X_i, define

$$T_i = \{(x_{i,j}, x_{i,j}, C_{i,t,j}) \mid j \in [n]\}$$

and

$$F_i = \{(x_{i,j}, x_{i,j+1}, C_{i,f,j}) \mid j \in [n]\},$$

where $x_{i,n+1} = x_{i,1}$. We set $C_{i,t,j} = c_j$ exactly if X_i occurs in C_j, and $C_{i,f,j} = c_j$ exactly if $\neg X_i$ occurs in C_j. Otherwise, the corresponding value is $\#$. Intuitively, T_i represents setting X_i to true and F_i represents setting it to false.

Set

$$A_1 = \bigcup_{i \in [m]} (T_i \cup F_i) \cup \{(\#, \#, \#)\}, \qquad k_{A_1} = mn + 1,$$

and

$$I = \{(x_{i,j}, x_{i,j}, \#) \mid i \in [m], j \in [n]\} \cup \{(\#, \#, c_j) \mid j \in [n]\}.$$

The three coordinates now have distinct purposes. The first forces one choice for every pair (i, j). The second enforces that these local choices are consistent for a fixed variable. The third records which clauses are satisfied. Covering the first coordinates requires at least one vector for every pair (i, j), while the clause targets force the additional vector $(\#, \#, \#)$. Hence, the complete budget is exhausted.

The second coordinates prevent mixing vectors from T_i and F_i. Any mixed selection repeats some $x_{i,j}$ and consequently leaves another $x_{i,\ell}$ uncovered. Thus, every VE-cover selects either all of T_i or all of F_i, which defines a consistent truth assignment. Finally, c_j occurs in the third coordinate of a selected vector exactly when the corresponding assignment satisfies clause C_j. Therefore, I is VE-covered if and only if φ is satisfiable. See Appendix A.2.3 for the detailed proof. Here, consider the following illustration of the key concept:

Example 9. *The cyclic shift is the key part of the construction. For the first variable X_1 of Example 7, we obtain*

$$T_1 = \{(x_{1,1}, x_{1,1}, c_1), (x_{1,2}, x_{1,2}, \#)\}$$

to represent setting the positive literal X_1 to true, and

$$F_1 = \{(x_{1,1}, x_{1,2}, \#), (x_{1,2}, x_{1,1}, c_2)\}.$$

to represent setting $\neg X_1$ to true. Selecting the first vector from T_1 and the second vector from F_1 would place $x_{1,1}$ twice in the second coordinate and leave $x_{1,2}$ uncovered. The target therefore rules out this mixed choice. The same cyclic argument works for arbitrary n.

Proposition 10. *Let $P \subseteq \{\delta, \xi\}$. Then, P-*`Inbetween-Decision` *is NP-complete.*

The final reduction uses the same three-dimensional gadget, but distributes its vectors over clusters of constant size. For every $i \in [m]$ and $j \in [n]$, let

$$A_{i,j} = \{(x_{i,j}, x_{i,j}, C_{i,t,j}), (x_{i,j}, x_{i,j+1}, C_{i,f,j})\}$$

with $k_{A_{i,j}} = 1$, again using cyclic indices. We additionally use $A_{\#} = \{(\#, \#, \#)\}$ with $k_{A_{\#}} = 1$ and the same target cluster I as above. Thus, $d = 3$ and every input cluster contains at most two vectors.

This modification replaces the global selection budget of the previous construction by local binary choices. Since $x_{i,j}$ occurs in the first coordinate only in $A_{i,j}$, every VE-cover must choose one vector from each such cluster. The same second-coordinate argument as above prevents mixing the true and false choices for a fixed variable. Hence, every VE-cover again defines a consistent truth assignment. The clause symbols in the third coordinate are covered exactly when all clauses are satisfied. This proves the equivalence. The full argument is given in Appendix A.2.4 together with an extended example.

The four propositions together provide the complete NP-hard side of the classification. All four constructions have polynomial size and can be computed in polynomial time, as detailed in the respective appendix proofs.

Theorem 11. *Let*

$$S \in \{\{\sigma, \nu, \tau\}, \{\sigma, \xi, \tau\}, \{\delta, \nu\}, \{\delta, \xi\}\}.$$

For every parameter prefix $P \subseteq S$, P-`Inbetween-Decision` *is NP-complete.*

Proof. This follows from Proposition 3, Proposition 5, Proposition 6, Proposition 8, and Proposition 10. □

3.2 Polynomial-Time Tractable Cases

The hardness results above show that the required information can be encoded either in the dimensionality or in the alphabet. We now identify the complementary parameter combinations that make the problem tractable. Again, full formal proofs are given in Appendix A (detailed subsections mentioned later, respectively).

Before we continue with the polynomial-time cases, we recall some standard terminology from parameterized complexity theory (for general references, see, e.g., Downey & Fellows (1999); Flum & Grohe (2006); Cygan et al. (2015)). A parameterized decision problem with parameter κ is called *fixed-parameter tractable* (FPT) if there exists a computable function f, a constant $c \in \mathbb{N}$, and an algorithm deciding the problem in time

$$f(\kappa(x)) \cdot |x|^c$$

on every input x. In the following, our parameters are tuples of the previously introduced quantities, such as (σ, δ), (ν, ξ), or (δ, τ). If the polynomial dependence on the input size is linear, i.e., if the runtime is of the form

$$f(\kappa(x)) \cdot |x|,$$

we say that the problem admits a *fixed-parameter linear-time* algorithm with respect to the considered parameter. This terminology is only used to highlight the stronger runtime behaviour of some of the polynomial-time algorithms below. The main classification result itself, however, only distinguishes polynomial-time decidability from NP-hardness. We continue with the first result.

Proposition 12. *Let $P \supseteq \{\sigma, \delta\}$ be any parameter prefix containing σ and δ. Then, P-*`Inbetween-Decision` *is in P. In particular,* `Inbetween-Decision` *is fixed parameter linear (FPL) with respect to the alphabet size together with the maximum dimensionality of data-points.*

Indeed, the key observation is that bounding both the alphabet size and the dimensionality bounds the number of structurally different vectors. Let $|\Sigma| \leq s$ and let the dimensionality be at most d. Then, there are at most

$$V = s^d$$

distinct vectors. Moreover, the target can require at most s different symbols in each of its d coordinates. Hence, there are at most sd coordinate-symbol requirements to cover and, consequently, at most

$$C = 2^{sd}$$

different states describing which of these requirements have already been covered.

This allows us to process the input clusters one after another using dynamic programming. Starting with the empty covering state, for every cluster A_i we consider every currently reachable state and every subset $A_i' \subseteq A_i$ with $|A_i'| \leq k_{A_i}$. We then store the state obtained by additionally covering all coordinate-symbol requirements supplied by A_i'. Since $|A_i| \leq V$, there are at most 2^V subsets to consider for each cluster. Both the number of states and the number of transitions per state therefore depend only on s and d. In other words, the algorithm never has to remember which vectors were selected. It only remembers which of the constantly many required elements of the target have already been covered.

After processing all clusters, the instance is accepted exactly if the state covering all requirements of I is reachable. Thus, the algorithm runs in time $O(f(s, d) \cdot n)$ for some computable function f. In particular, `Inbetween-Decision` is fixed-parameter linear with respect to the alphabet size together with the dimensionality. The full proof is given in Appendix A.3.1. We continue with the next case.

Proposition 13. *Let $P \supseteq \{\nu, \xi\}$ be any parameter prefix containing ν and ξ. Then, P-*`Inbetween-Decision` *is in P. In particular,* `Inbetween-Decision` *is fixed parameter tractable (FPT) with respect to the number of input clusters together with the maximum number of elements per input cluster.*

This case becomes tractable for a different reason. If the number of input clusters is bounded by ν and every cluster contains at most ξ vectors, then each cluster has at most 2^ξ possible subsets. Hence, there are at most

$$(2^\xi)^\nu = 2^{\nu\xi}$$

possible selections across all clusters. This number depends only on the two bounded parameters.

We can therefore enumerate all selections satisfying the respective bounds k_{A_i} and test for each of them whether their union VE-covers I. The latter test is polynomial in the input size. Consequently, `Inbetween-Decision` is fixed-parameter tractable with respect to the number of clusters together with the maximum cluster size. See Appendix A.3.2 for the full analysis and formal proof.

Proposition 14. *Let $P \supseteq \{\delta, \tau\}$ be any parameter prefix containing δ and τ. Then, P-`Inbetween-Decision` is in P. In particular, `Inbetween-Decision` is fixed-parameter linear with respect to the dimensionality together with the target-cluster size.*

The last case is less immediate. The alphabet and the input clusters may now be arbitrarily large. However, if the dimensionality is bounded by d and the target contains at most t vectors, then only constantly many coordinate-symbol pairs can ever be relevant for covering I.

For every coordinate $\ell \in [d]$, let

$$D_\ell = \{p[\ell] \mid p \in I\}$$

be the set of symbols required by the target in that coordinate. Clearly, $|D_\ell| \leq t$. Introduce a fresh dummy symbol $\#$ and map every input vector p to its relevant projection $\rho(p)$ defined by

$$\rho(p)[\ell] = \begin{cases} p[\ell], & \text{if } p[\ell] \in D_\ell, \\ \#, & \text{otherwise.} \end{cases}$$

All symbols outside D_ℓ are equivalent with respect to covering I in coordinate ℓ, since none of them can cover a target requirement there.

Example 15. *For example, if $I = \{AC, GT\}$, then $D_1 = \{A, G\}$ and $D_2 = \{C, T\}$. Hence, both CC and TC project to $(\#, C)$. Although distinct input vectors, they are indistinguishable with respect to VE-covering I.*

For each cluster, we therefore only have to keep the distinct projected vectors

$$B_i = \{\rho(p) \mid p \in A_i\}.$$

Since every coordinate has at most t relevant symbols together with the dummy symbol, there are at most

$$V = (t+1)^d$$

different projected vectors. Thus, every potentially huge input cluster is reduced to constantly many relevant behaviours.

We can now apply the same dynamic-programming idea as in Proposition 12. There are at most dt target requirements and therefore at most 2^{dt} partial covering states. For each cluster, only subsets of the constantly sized set B_i have to be considered. The projection can be computed by a single scan over all input vectors, and the subsequent dynamic program performs only parameter-dependent work per cluster. This yields a running time

$$O\left(f(d,t) \cdot \left(n + \sum_{i=1}^{n} |A_i|\right)\right)$$

for some computable function f. The complete correctness and runtime arguments are given in Appendix A.3.3.

This concludes all cases. The three algorithms exploit three different sources of bounded complexity. Bounding σ and δ leaves only constantly many vector types and covering states. Bounding ν and ξ leaves only constantly many possible selections. Finally, bounding δ and τ leaves only constantly many target-relevant vector types after projection. Together, these cases form the complete polynomial-time side of the dichotomy.

Theorem 16 (Dichotomy)**.** *Let $P \subseteq \{\sigma, \delta, \nu, \xi, \tau\}$. Then, P-`Inbetween-Decision` is decidable in polynomial time if and only if*

$$\{\sigma, \delta\} \subseteq P, \qquad \{\nu, \xi\} \subseteq P, \qquad \text{or} \qquad \{\delta, \tau\} \subseteq P.$$

Otherwise, it is NP-complete.

Proof. The polynomial-time cases follow from Proposition 12, Proposition 13, and Proposition 14. All remaining cases are NP-complete by Theorem 11 and Proposition 3. □

The complete boundary is illustrated in Figure 2.

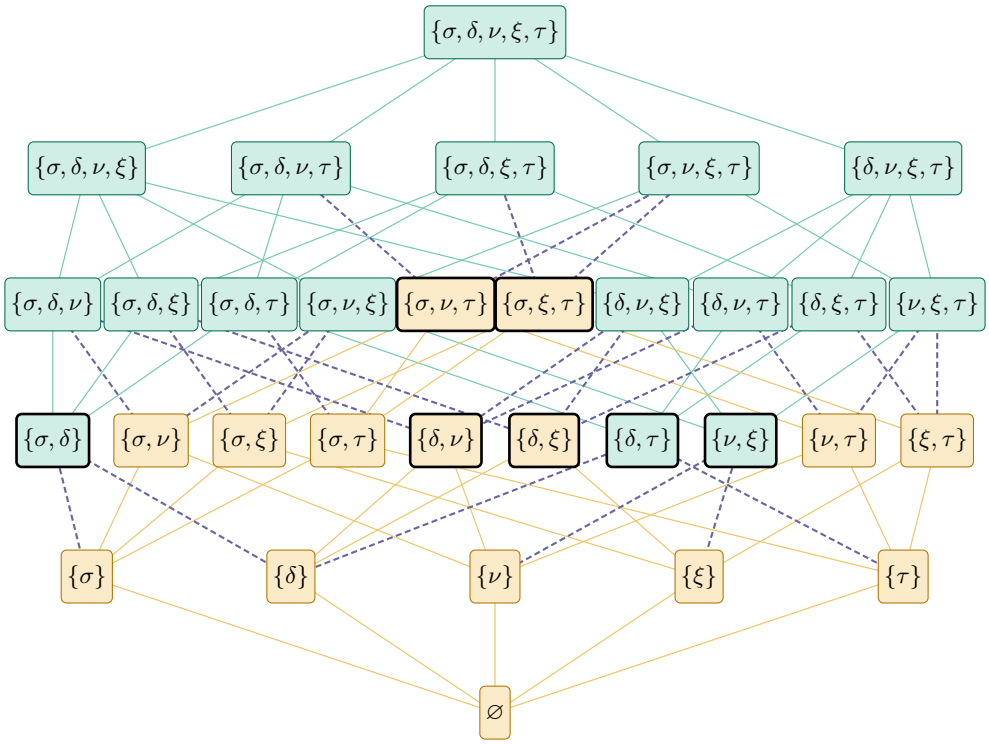

Figure 2: Complexity dichotomy over the parameter subsets of $\{\sigma, \delta, \nu, \xi, \tau\}$. Cyan colored blocks mark the cases that are in P, orange colored ones the cases that are NP-hard. Dotted edges mark the border between both areas. Vertices with thick borders mark the cases considered explicitly above, from which the remaining cases follow

4 Conclusion

We introduced `Inbetween-Decision`, an exact symbolic formulation of in-betweenness in which a target is reconstructed coordinate-wise from a budgeted selection of actual cluster members. We obtained a complete complexity dichotomy with respect to the five structural parameters σ, δ, ν, ξ, and τ. Among all 2^5 parameter regimes, the problem is decidable in polynomial time exactly when at least one of $\{\sigma, \delta\}$, $\{\nu, \xi\}$, or $\{\delta, \tau\}$ is bounded. All remaining cases are NP-complete. Moreover, all three polynomial-time cases are fixed-parameter tractable, while the cases parameterized by (σ, δ) and (δ, τ) even admit fixed-parameter linear-time algorithms.

Besides providing a complete classification, the dichotomy also gives a structural guideline for the algorithmic treatment of different instances. Whenever the parameters in one of the three tractable combinations are small, exact in-between detection can be approached by polynomial-time algorithms, leaving room for improving the respective parameter dependence and for developing practically efficient implementations. If none of these structural restrictions applies, then, unless $\mathrm{P} = \mathrm{NP}$, no polynomial-time exact algorithm can be expected in general. For these cases, approximation algorithms, relaxed or noisy variants of VE-covering, heuristics, average-case approaches, or additional parameterizations provide natural directions for further investigation.

At the same time, the obtained NP-hardness results concern worst-case complexity and therefore do not imply that all practically occurring instances are difficult, nor do they establish inapproximability. Similarly, fixed-parameter tractability alone does not guarantee practical efficiency for large parameter values. Our current formulation furthermore deliberately focuses on exact reconstruction over aligned symbolic features. Extending the model towards approximate or robust reconstruction, handling uncertain or missing feature values, and establishing connections to continuous notions of in-betweenness therefore remains an interesting direction for future work.

AI USE STATEMENT

In this work, we have not used generative AI tools for any of the tasks with required disclosure, in particular, it was not used for help in developing theoretical models or conceptual frameworks, for the formulation of mathematical claims, for providing critical ingredients for proving mathematical claims, and for assistance in writing the proofs. The tasks of generating synthetic data sets, proposing or refining a hypothesis, designing or providing feedback on reserach methodology and experiments, implementing methods, assistance with translation, clearning and reformating datasets, supporting qualitative and thematic data analysis, and interpreting results are not applicable to this work. Additionally, we used generative AI tools for providing initial sketches of some of the provided figures, to obtain some initial feedback regarding the obtained theoretical results, to identify some of the relevant literature, edit some parts of the paper to improve readability, and for proofreading some of the textual parts of the paper. We have reviewed all AI-assisted work. In particular, we reviewed and edited proposed changes to the main text before implementing such changes manually, reviewed the recommended literature for relevance, and we checked in detail whether the LLM-generated feedback for the theoretical results was indeed valid. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

For reproducibility of the theoretical results provided in this paper, extensive, full and formal proofs for all results in the paper can be found in the appendix. For intuition, for all NP-hardness results, the appendix contains extended examples illustrating the constructions that are used in the respective proofs.

REFERENCES

James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981. doi: 10.1007/978-1-4757-0450-1.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

Chaofan Chen, Oscar Li, Chaofan Tao, Alina Jade Barnett, Jonathan Su, and Cynthia Rudin. This looks like that: Deep learning for interpretable image recognition. *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, 2018. doi: 10.48550/arxiv.1806.10574.

Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi: 10.1007/978-3-319-21275-3.

Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999. doi: 10.1007/978-1-4612-0515-9.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 226–231, 1996.

Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. doi: 10.1007/3-540-29953-X.

Andrea Göhring, Mirjam Bayer, Daniyal Kazempour, Sweety Mohanty, and Claudius Zelenka. Data fusion between land and sea: Multi-isotope fingerprints of viking animals and modern plants. In *2024 25th IEEE International Conference on Mobile Data Management (MDM)*, pp. 319–324. IEEE, 2024.

Andrea Göhring, Daniyal Kazempour, Jan Robert Bensien, Sweety Mohanty, Claudius Zelenka, Matthias Renz, Peer Kröger, Gisela Grupe, and Amei Lang. Provenance analysis based on cluster in-betweenness and support vector machines: Identifying migrant candidates using multi-isotope fingerprints. *Archaeometry*, 2025.

Jens Gramm, Rolf Niedermeier, and Peter Rossmanith. Fixed-parameter algorithms for CLOSEST STRING and related problems. *Algorithmica*, 37(1):25–42, 2003. doi: 10.1007/s00453-003-1028-3.

Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher (eds.), *Complexity of Computer Computations*, pp. 85–103. Plenum Press, 1972. doi: 10.1007/978-1-4684-2001-2_9.

Daniyal Kazempour, Claudius Zelenka, and Peer Kröger. The missing link? on the in-between instance detection task. In *EDBT*, pp. 359–364, 2024.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. *arXiv*, 2020. doi: 10.48550/arxiv.2007.04612.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pp. 4765–4774, 2017.

Geoffrey J. McLachlan and David Peel. *Finite Mixture Models*. Wiley, New York, 2000. doi: 10.1002/0471721182.

Lukas Ruff, Jacob R. Kauffmann, Robert A. Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G. Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021. doi: 10.1109/JPROC.2021.3052449.

Eike M Sch, Daniyal Kazempour, Peer Kr, Natascha Oppelt, et al. Correcting wave-induced reflectance anomalies in pan-sharpened very high-resolution satellite data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2026.

A Additional Content for Section 3

A.1 Preliminary Results and Notes

Remark 17. *We allow a set of targets because in-between objects often come in groups, e.g., several migrant candidates in provenance analysis (Göhring et al. (2025)). The problem captures the reconstruction core of in-betweenness, an object is in-between when, in addition, no single cluster can reconstruct it on its own, so that at least two clusters must contribute.*

Example 18. *Let $\Sigma = \mathbb{N}$ and assume a dimensionality of $d = 2$. Then, every vector is just a pair. Let $A_1 = \{(1,2),(1,4),(3,2)\}$ and $A_2 = \{(2,3),(3,4),(4,1)\}$. Set $k_{A_1} = 2$ and $k_{A_2} = 2$. Finally, set $I = \{(1,1),(2,2),(3,3),(4,4)\}$. Then, the answer to* `Inbetween-Decision` *with respect to these inputs is* yes*, as we can take $A'_1 = \{(1,4),(3,2)\}$ and $A'_2 = \{(2,3),(4,1)\}$ and VE-cover I with $A'_1 \cup A'_2$.*

A.1.1 Proof of Proposition 3

Proof. We show that there exists a polynomial-time verifier for the problem. For that, assume an arbitrary input instance of P-`Inbetween-Decision` for some prefix $P \subseteq \{\sigma, \delta, \nu, \xi, \tau\}$. As certificate, we guess, for each input cluster A_i, with $i \in [n]$, a subset $A'_i \subseteq A_i$. To encode this certificate, it suffices to list the indices of the selected vectors in each cluster. Hence, the size of the certificate is at most linear in the total number of vectors occurring in the input and, therefore, polynomially bounded in the input size.

It remains to argue that the certificate can be verified in polynomial time. First, for each $i \in [n]$, we check whether all guessed indices indeed refer to elements from A_i and whether $|A'_i| \leq k_{A_i}$. This can clearly be done in polynomial time. Afterwards, we check whether $\bigcup_{i=1}^{n} A'_i$ VE-covers I. For that, we iterate over all vectors $p \in I$ and all coordinates $\ell \in [d]$. For each such pair (p, ℓ), we check whether there exists some selected vector

$$v \in \bigcup_{i=1}^{n} A'_i$$

such that $v[\ell] = p[\ell]$. If such a vector exists for every $p \in I$ and every $\ell \in [d]$, then the guessed subsets form a VE-covering of I and the verifier accepts. Otherwise, it rejects.

The described verification procedure takes time at most

$$\mathcal{O}\left(|I| \cdot d \cdot \sum_{i=1}^{n} |A_i'|\right)$$

after checking the validity of the guessed indices and the cardinality bounds. Since $\sum_{i=1}^{n} |A_i'| \leq \sum_{i=1}^{n} |A_i|$, this is polynomial in the size of the input. Hence, the verifier runs in polynomial time.

Finally, the verifier accepts if and only if there exist subsets $A_i' \subseteq A_i$, for all $i \in [n]$, with $|A_i'| \leq k_{A_i}$ such that $\bigcup_{i=1}^{n} A_i'$ VE-covers I. This is exactly the definition of a yes-instance of P-`Inbetween-Decision`. Thus, P-`Inbetween-Decision` is in NP. □

A.2 NP-Hardness Results

A.2.1 Proof and Example for Proposition 5

Proof. NP-containment follows from Proposition 3. We continue with the proof of NP-hardness. For that, we provide a polynomial-time many-one reduction from the 3-CNF-SAT problem. An instance of 3-CNF-SAT is a formula $\varphi = \{C_1, ..., C_n\}$ that consists of a conjunction of clauses C_1 to C_n, for some $n \in \mathbb{N}$, where each clause $C_i = \{X_{i,1}, X_{i,2}, X_{i,3}\}$, for $i \in [n]$, is a disjunction of three literals $X_{i,1}$, $X_{i,2}$, and $X_{i,3}$, where each $X_{i,j}$, for $j \in [3]$, is either a propositional variable $X_{i,j} = X_\ell \in \mathcal{X}$ or its negated counterpart $X_{i,j} = \overline{X_\ell} \in \overline{\mathcal{X}}$. We say that φ returns true if there exists a satisfying assignment of variables for φ, otherwise the instance returns false.

Let φ be an arbitrary instance of 3-CNF-SAT. Let $P \subseteq \{\sigma, \nu, \tau\}$ be any parameter prefix not containing δ and ξ. We construct an instance of P-`Inbetween-Decision` in the following way. Let m be the total number of distinct variables occurring in φ and let n be its number of clauses. We set $\Sigma = \{0, 1\}$ (alphabet size constant bound of 2), we assume only one cluster A_1 (constant bound of 1 cluster), and we assume that the target cluster I has only one element (constant bound of $|I| = 1$). Now, we set the dimensionality of the considered vectors to $d = m + n$. The only vector in I is the $(m + n)$-dimensional data point $p_I \in \{1\}^{m+n}$ only containing 1s. Next, for each variable $X_i \in \mathcal{X}$ that occurs somewhere in φ, we construct two $(m + n)$-dimensional data points p_{x_i} and $p_{\overline{x}_i}$ in the cluster A_1 in the following manner: First, we set $p_{x_i}[i] = p_{\overline{x}_i}[i] = 1$. Next, for each $j \in [m]$ with $j \neq i$ we set $p_{x_i}[j] = p_{\overline{x}_i}[j] = 0$. Finally, for each clause C_j, with $j \in [n]$, that contains X_i, we set $p_{x_i}[m+j] = 1$. Analogously, for all clauses C_j, with $j \in [n]$, that contain $\overline{X}_i$, we set $p_{\overline{x}_i}[m+j] = 1$. All other values in both data points we set to 0. That way, we end up with $2m$ many elements in A_1. Finally we set $k_{A_1} = m$ to the number of variables in φ. This concludes the construction.

Notice that only the dimensionality of the vectors and the number of elements in A_1 have a variable size while all other parts of the construction can be bound by a constant independently of the input φ. Also, notice that the construction can be done in polynomial time $O(m(m + n))$ as we have to construct $2m + 1$ many vectors, each of which has dimensionality $m + n$. For the last n dimensions of each vector we have to check whether a given variable (or its negated form) equals one of three literals, which is a constant bound.

($\Rightarrow$): Assume there exists a satisfying assignment $\beta : \mathcal{X} \to \{0, 1\}$ such that $[\![\varphi]\!]_\beta = 1$. We select a subset $A_1' \subseteq A_1$ consisting of exactly m elements from A_1 in the following way: For each $i \in [m]$, if $\beta(X_i) = 1$ then select p_{x_i}. Otherwise, we know $[\![\overline{X}_i]\!]_\beta = 1$, hence, we select $p_{\overline{x}_i}$. By that, we select exactly $k_{A_1} = m$ many elements. As we select exactly one vector per distinct variable X_i, we know that the first m elements of the target vector in I are covered. In addition, we know that β is a satisfying assignment, implying that every clause has at least one literal set to true. By that, as we select exactly those vectors corresponding to the literals that are set to true, we know that for each clause C_i, there must exist a vector p_{x_i} or $p_{\overline{x}_i}$ that sets the dimension $m + i$ to 1. Hence, we also cover all other dimensions of p_I using the selected data-points. Therefore, A_1' is a VE-covering of I, concluding this direction.

($\Leftarrow$): Now, assume there exists a VE-covering of I, expressed by a set A_1' with at most $k_{A_1} = m$ many elements. By the constraint that we have to cover the first m elements of p_I with a 1, we

know that A'_1 has exactly m elements and, for each $i \in [m]$, we know that either p_{x_i} or $p_{\overline{x}_i}$ has to be selected. We construct an assignment of variables β in the following way: For each selected $p_{x_i} \in A'_1$, we set $\beta(X_i) = 1$. Otherwise, for each selected $p_{\overline{x}_i} \in A'_1$, we set $\beta(X_i) = 0$, resulting in $[\![\overline{X}_i]\!]_\beta = 1$. Now, as A'_1 VE-covers I, we know that, for each $j \in [n]$, the value $p_I[m+j] = 1$ is also covered. So, for each clause $C_j \in \varphi$, there exists at least one literal $X_{j,\ell}$ that corresponds to either a variable $X_i \in \mathcal{X}$ or its negated counterpart $\overline{X}_i$ that is set to 1 by β. Hence, β is a satisfying assignment of φ, concluding the reduction. □

Example 19. *Consider the following propositional formula as part of a 3-CNF-SAT instance.*

$$(X_1 \vee X_2 \vee \neg X_3) \wedge (\neg X_1 \vee X_2 \vee X_4)$$

Using set notation, we can also write this instance as

$$\varphi = \{\{X_1, X_2, \neg X_3\}, \{\neg X_1, X_2, X_4\}\}.$$

This conjunction consists of two clauses $C_1 = \{X_1, X_2, \neg X_3\}$ and $C_2 = \{\neg X_1, X_2, X_4\}$, where each element of each clause is a literal. As this instance consists of $m = 4$ distinct variables and $n = 2$ distinct clauses, we obtain from the construction of the corresponding `Inbetween-Decision` *instance a dimensionality of $d = 2 + 4 = 6$. Hence, the target set I consists of the following element:*

$$I = \left\{ \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \right\}$$

By the construction, we only construct one input cluster A_1 which contains two vectors per variable, one for each, the negated and the non-negated literal. The vector for each literal for a variable X_i marks in the ith component which variable is targeted and in the $(m+j)^{\text{th}}$ component, for $j \in [n]$, which clause is set to true by that literal. We obtain the following set (the horizontal line in each vector is just an indicator representing which dimensions correspond to a variable selection and which dimensions correspond to the truth values of each clause):

$$A_1 = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \hline 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \hline 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \hline 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \hline 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \hline 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \hline 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \hline 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \hline 0 \\ 0 \end{pmatrix} \right\}$$

By setting $k_{A_1} = 4$, we force the selection of one literal vector per variable (either setting the negated or the non-negated version to true). For this instance, we see that we can select the following subset to VE-cover the single vector in I, highlighting that the input 3-CNF-SAT instance is indeed satisfiable (note that this selection corresponds to mapping all four variables to 1 in the input 3-CNF-SAT instance):

$$A'_1 = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \hline 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \hline 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \hline 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \hline 0 \\ 1 \end{pmatrix} \right\} \subset A_1$$

A.2.2 Proof and Example for Proposition 6

Proof. As before, containment in NP follows from Proposition 3. For NP-hardness, we again provide a polynomial-time many-one reduction. For the definition of that problem setting, please see the proof of Proposition 5.

Assume $\varphi = \{C_1, ..., C_n\}$ to be an arbitrary instance of 3-CNF-SAT with variables $\{X_1, ..., X_m\}$ such that, for each $i \in [n]$, we have $C_i = \{X_{i,1}, X_{i,2}, X_{i,3}\}$, where $X_{i,j} \in \{X_1, \overline{X}_1, ..., X_m, \overline{X}_m\}$.

Let $P \subseteq \{\sigma, \xi, \tau\}$ be any parameter prefix not containing δ and ν. Notice that the primary difference to the result from Proposition 5 lies in the fact that we can now have an arbitrary number of clusters in our input but each cluster must have a number of elements smaller than some given constant bound. The basic idea is similar to before: We use the values of the data-points to encode which clause is satisfied by a given selected variable. As we can use distinct clusters to encode distinct variables, however, in comparison to before, we can use data points with a smaller dimensionality than in Proposition 5. To be precise, we construct an instance of P-`Inbetween-Decision` as follows: Set $\Sigma = \{0, 1\}$ (constant bound of 2 for the alphabet size). We define $d = n$ as the dimensionality of the data-points considered. For each variable X_i, with $i \in [m]$, we add a d-cluster A_i that consists of two d-dimensional data-points p_i and $\overline{p}_i$ (constant bound of 2 elements per cluster). For each $i \in [m]$ and for each $j \in [n]$, if in clause C_j there exists a literal $X_{j,\ell}$, with $\ell \in \{1, 2, 3\}$, such that $X_i = X_{j,\ell}$, then we set $p_i[j] = 1$, otherwise we set $p_i[j] = 0$ (Analogously, if $\overline{X}_i = X_{j,\ell}$ then set $\overline{p}_i[j] = 1$, otherwise set $\overline{p}_i[j] = 0$). Hence, each vector encodes which clauses become true if a certain variable or its negated form gets set to true. For each cluster A_i, with $i \in [m]$, we add a selection bound of $k_{A_i} = 1$, allowing only for either the encoded positive variable or the negated variant to be chosen. Finally, we add one element to the target set I (constant bound of 1 for $|I|$), the d-dimensional all one vector $p_I \in \{1\}^d$. This concludes the construction.

Notice that only the dimensionality of each vector d and the number of clusters have a variable size while all other parameters of this instance's construction are bound by a constant value that is independent of the input formula. The polynomial time bound of $O(m \cdot n)$ follows from the fact that we add one cluster per variable, each containing two elements per variable, resulting in $2m$ input vectors and one target vector to be constructed in total. Each vector has a dimensionality of n and for each dimension we have to check whether one of three given literals equals a given variable or its negated form.

($\Rightarrow$): Assume there exists a satisfying assignment $\beta : \{X_1, ..., X_m\} \to \{0, 1\}$ with $[\![\varphi]\!]_\beta = 1$. For each $i \in [m]$, we select $A'_i = \{p_i\} \subset A_i$ if $\beta(X_i) = 1$. Otherwise, we select $A'_i = \{\overline{p}_i\}$. Let $A' = \bigcup_{i=1}^{m} A'_i$. For each clause C_j, with $j \in [n]$, we know that there exists a literal $X_{j,\ell}$, with $\ell \in \{1, 2, 3\}$, such that $\beta(X_{j,\ell}) = 1$. Assume w.l.o.g. that $X_{j,\ell} = X_i$, for some $i \in [m]$ (the following holds analogously for the case $X_{j,\ell} = \overline{X}_i$). Then, we know that $p_i \in A'_i$ and therefore $p_i \in A'$ and we also know that $p_i[j] = 1$, as X_i occurs in clause C_j. As we can do that for every clause, we know that A' VE-covers I. As all selection bounds k_{A_i} are respected, we found a solution for the constructed instance, concluding this direction of the proof.

($\Leftarrow$): Now assume we can select subsets $A'_i \subset A_i$ with $|A'_i| \leq 1$ such that $\bigcup_{i=1}^{m} A'_i = A'$ VE-covers I. For each $i \in [m]$, due to the bound k_{A_i}, we know that either $p_i \in A'$, $\overline{p}_i \in A'$, or $p_i, \overline{p}_i \notin A'$, but not $p_i, \overline{p}_i \in A'$. We construct an assignment of variables β such that $\beta(X_i) = 1$ if $p_i \in A'$. Otherwise, we set $\beta(X_i) = 0$, resulting in $[\![\overline{X}_i]\!]_\beta = 1$. Now, for each $j \in [n]$, we know that there exists a vector $p \in A'$ such that $p[j] = 1$. By construction β sets the literal corresponding to the data-point p to 1. Therefore, as we can do that for all clauses in φ, we know that β sets all clauses to true, resulting in $[\![\varphi]\!]_\beta = 1$, making it a satisfying assignment. This concludes the reduction. □

Example 20. *For comparison purposes, we consider the same* 3*-CNF-SAT formula as in Example 19 which is represented by the following:*

$$(X_1 \vee X_2 \vee \neg X_3) \wedge (\neg X_1 \vee X_2 \vee X_4)$$

Note that we have $m = 4$ *distinct variables and* $n = 2$ *distinct clauses. By the construction given in Proposition 6, we construct* 4 *input clusters* A_1 *to* A_4*, one for each variable, that each contain two vectors, each for representing whether the negated or the non-negated literal are set to be true. Similar to Example 19, the target set* I *only consists of* 1 *vector that only contains* 1*s. The dimensionality, however, this time corresponds to the number of clauses and not to the number of clauses plus the number of variables, resulting in two-dimensional vectors for this instance. Finally, as given by the construction, we set* $k_{A_1} = k_{A_2} = k_{A_3} = k_{A_4} = 1$*, allowing only for the selection of* 1 *vector per input cluster to cover the potential in-between instance. We obtain the following target vector:*

$$I = \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}$$

In addition, we obtain the following 4 *input clusters.*

$$A_1 = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\} \quad A_2 = \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right\} \quad A_3 = \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\} \quad A_4 = \left\{ \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right\}$$

As mentioned before, each dimension corresponds to a clause where a 1 *represents that setting the corresponding literal to* 1 *sets the corresponding clause to* 1 *as well while a* 0 *represents that the corresponding clause remains false. For example, selecting the first vector from* A_1 *represents setting* X_1 *to true while setting* $\neg X_1$ *to false. This results in the first clause being set to true while the second clause is not set to true by that selection. Similarly to Example 19, also setting* X_4 *to true, hence selecting the first vector from the* A_4*, also provides us with a* 1 *in the second dimension, representing the second clause of the* 3*-CNF-SAT formula being set to* 1 *as well. With that choice, it does not matter which of the remaining vectors in* A_2 *and* A_3 *we select as the vector from* I *is already covered (representing that clause one and clause two of the* 3*-CNF-SAT formula are already set to true).*

A.2.3 Proof and Example for Proposition 8

Proof. The containment in NP follows from Proposition 3. As in the previous results, we obtain NP-hardness by a reduction from 3-CNF-SAT. Let $\varphi = \{C_1, ..., C_n\}$ be a formula in 3-conjunctive normal form with variables $\{X_1, ..., X_m\}$ such that $C_i = \{X_{i,1}, X_{i,2}, X_{i,3}\}$ for literals $X_{i,j} \in \{X_1, \overline{X}_1, ..., X_m, \overline{X}_m\}$. We assume a constant bound of $d = 3$ dimensions per data-point and consider a single input cluster A_1, setting the number of input clusters to the constant 1. For each variable, we add $2n$ data-points to A_1. These data points represent two cycles of length n, corresponding to two possible consistent selections of vectors (see Example 21 afterwards for an intuitive visualization). To be precise, for each $i \in [m]$, we first add the symbols $x_{i,1}, ..., x_{i,n} \in \Sigma$ to the alphabet (notice that this time we are allowed an unbounded number of input symbols). At this point, we also add the symbols $c_1, ..., c_n, \# \in \Sigma$ to the alphabet. These will be used to determine satisfied clauses later. Next, for a fixed variable X_i, we add the following two sets of vectors to A_1, first

$$\{(x_{i,1}, x_{i,1}, C_{i,t,1}), (x_{i,2}, x_{i,2}, C_{i,t,2}), ..., (x_{i,n}, x_{i,n}, C_{i,t,n})\}$$

and then

$$\{(x_{i,1}, x_{i,2}, C_{i,f,1}), (x_{i,2}, x_{i,3}, C_{i,f,2}), ..., (x_{i,n}, x_{i,1}, C_{i,f,n})\}.$$

For each $j \in [n]$, we set $C_{i,t,j} = c_j$ if X_i occurs as a literal in clause C_j. If not, we set $C_{i,t,j} = \#$. On the other hand, if $\overline{X}_i$ occurs in clause C_j, we set $C_{i,f,j} = c_j$. Otherwise, we set $C_{i,f,j} = \#$. In addition to that, to the target cluster I, we add, for each $i \in [m]$ and $j \in [n]$ the vector $(x_{i,j}, x_{i,j}, \#)$ and generally, for each $j \in [n]$, we add the vector $(\#, \#, c_j)$. Finally, we add the vector $(\#, \#, \#)$ to A_1 and we set $k_{A_1} = m \cdot n + 1$.

So, what is the intuition behind this approach? Each of the two subsets of vectors stated before corresponds to setting the variable X_i to either true or to false. As we need all symbols $x_{i,j}$, for $i \in [m]$ and $j \in [n]$, to occur in coordinates 1 and 2 of selected vectors (by the construction of I), we see that this can only be achieved by selecting, for each variable X_i, at least n vectors from the constructed ones above. In addition to that, we have to select the $(\#, \#, \#)$ vector to map the $\#$ symbols occurring in the first two coordinates of the vectors containing c_j in I and to cover the third coordinate of the vectors containing $x_{i,j}$ in I. That directly results in us having to pick at least $m \cdot n + 1$ many vectors which is, in turn, also the bound k_{A_1}. Thus, we have to select exactly $m \cdot n + 1$ many vectors for potential VE-coverings of I leaving exactly n selected vectors for each variable.

It remains to argue that, for a fixed variable X_i, vectors from the two cycles cannot be mixed in a VE-cover. Consider the second coordinates of the selected vectors. If we choose the vector

$$(x_{i,j}, x_{i,j}, C_{i,t,j}),$$

then the selected vector contributes $x_{i,j}$ to the second coordinate. If we choose

$$(x_{i,j}, x_{i,j+1}, C_{i,f,j}),$$

then the selected vector contributes $x_{i,j+1}$ to the second coordinate (assume $x_{i,n+1} = x_{i,1}$). As the vectors $(x_{i,j}, x_{i,j}, \#)$ occur in I, all symbols $x_{i,1}, \ldots, x_{i,n}$ also have to occur in the second coordinate of selected vectors. Since exactly n vectors are selected for the variable X_i, this means that the selected second-coordinate values have to be exactly the set $\{x_{i,1}, \ldots, x_{i,n}\}$.

Assume now that we mix both choices. Then there exists some index $j \in [n]$ such that we choose the second-circle vector for index $j-1$ but the first-circle vector for index j, where the indices are again considered modulo n. But then both selected vectors contribute the same value $x_{i,j}$ in the second coordinate:

$$(x_{i,j-1}, x_{i,j}, C_{i,f,j-1}) \quad \text{and} \quad (x_{i,j}, x_{i,j}, C_{i,t,j}).$$

Consequently, some other value from $\{x_{i,1}, \ldots, x_{i,n}\}$ is missing in the second coordinate, which contradicts the fact that I has to be VE-covered. Therefore, no mixed choice is possible. Hence, for each variable X_i, the selected vectors are either exactly

$$\{(x_{i,1}, x_{i,1}, C_{i,t,1}), (x_{i,2}, x_{i,2}, C_{i,t,2}), \ldots, (x_{i,n}, x_{i,n}, C_{i,t,n})\}$$

or exactly

$$\{(x_{i,1}, x_{i,2}, C_{i,f,1}), (x_{i,2}, x_{i,3}, C_{i,f,2}), \ldots, (x_{i,n}, x_{i,1}, C_{i,f,n})\}.$$

Thus, selecting vectors that VE-cover I indeed corresponds to choosing one of the two possible truth values for each variable.

We next argue that the construction can be performed in polynomial time. The single input cluster contains $2mn + 1$ vectors of dimensionality 3. For every $C_{i,t,j}$ and $C_{i,f,j}$, we only have to check whether the corresponding literal occurs in clause C_j, which takes constant time since each clause contains three literals. The target cluster contains $mn + n$ vectors, namely one vector for every pair (i, j) and one additional vector for every clause. Hence, the overall construction takes $O(mn)$ time.

($\Rightarrow$): Assume there exists a satisfying assignment $\beta : \{X_1, ..., X_m\} \to \{0, 1\}$ of φ. We construct the selected subset A_1' of A_1 by the following: For each variable X_i, with $i \in [m]$, if $\beta(X_i) = 1$, we select the subset

$$\{(x_{i,1}, x_{i,1}, C_{i,t,1}), (x_{i,2}, x_{i,2}, C_{i,t,2}), ..., (x_{i,n}, x_{i,n}, C_{i,t,n})\}.$$

Otherwise, we know that $[\![\overline{X}_i]\!]_\beta = 1$. In that case, we select the subset

$$\{(x_{i,1}, x_{i,2}, C_{i,f,1}), (x_{i,2}, x_{i,3}, C_{i,f,2}), ..., (x_{i,n}, x_{i,1}, C_{i,f,n})\}.$$

Finally, we select the vector $(\#, \#, \#)$. By that, we have selected exactly $m \cdot n + 1 = k_{A_1}$ many vectors in $A_1' \subset A_1$. Now we have to conclude that A_1' is actually a VE-covering of I. All vectors $(x_{i,j}, x_{i,j}, \#)$ are clearly covered. For each clause C_j, with $j \in [n]$, there exists a literal in C_j that is satisfied by β. If this literal is X_i, then $\beta(X_i) = 1$ and, by construction, $C_{i,t,j} = c_j$. Hence, c_j occurs in the third coordinate of a selected vector from the true-cycle of X_i. Analogously, if the satisfied literal is $\neg X_i$, then $\beta(X_i) = 0$ and $C_{i,f,j} = c_j$, so c_j occurs in the third coordinate of a selected vector from the false-cycle of X_i. Therefore, every $(\#, \#, c_j) \in I$ is VE-covered.

($\Leftarrow$): Now, assume there exists a subset $A_1' \subset A_1$ with $|A_1'| \leq m \cdot n + 1 = k_{A_1}$ that VE-covers I. As established before, we know that we must actually have an equality $|A_1'| = m \cdot n + 1$. We also established in the arguments before that, for each variable X_i, with $i \in [m]$, we have to pick either the subset

$$\{(x_{i,1}, x_{i,1}, C_{i,t,1}), (x_{i,2}, x_{i,2}, C_{i,t,2}), ..., (x_{i,n}, x_{i,n}, C_{i,t,n})\}$$

or

$$\{(x_{i,1}, x_{i,2}, C_{i,f,1}), (x_{i,2}, x_{i,3}, C_{i,f,2}), ..., (x_{i,n}, x_{i,1}, C_{i,f,n})\}$$

to cover all vectors $(x_{i,1}, x_{i,1}, \#)$ to $(x_{i,n}, x_{i,n}, \#)$ in I. Also, we know that we have to pick the vector $(\#, \#, \#)$ as otherwise the symbols $\#$ in coordinates 1 and 2 are unachievable. By that, we already picked exactly $m \cdot n + 1$ many vectors, allowing for nothing more. As A_1' is a VE-covering of I, we know that the values c_1 to c_n must occur in the third coordinates of some of the selected vectors. We construct an assignment $\beta : \{X_1, ..., X_m\} \to \{0, 1\}$ as follows: For each $i \in [m]$, if the subset

$$\{(x_{i,1}, x_{i,1}, C_{i,t,1}), (x_{i,2}, x_{i,2}, C_{i,t,2}), ..., (x_{i,n}, x_{i,n}, C_{i,t,n})\}$$

was chosen as part of A_1', then we set $\beta(X_i) = 1$. Otherwise, if the other subset was chosen, we set $\beta(X_i) = 0$, resulting in $[\![\overline{X_i}]\!]_\beta = 1$. Now, for each c_j, with $j \in [n]$, we know that there exists some $i \in [m]$ such that c_j occurs in a selected vector $(x_{i,j}, x_{i,a}, c_j)$, where $a = j$ or $a = j + 1$, with indices considered modulo n. By the construction of β, the literal represented by this selected vector is satisfied. By the construction of A_1, we know that the corresponding literal must occur in clause C_j as we set the value of dimension 3 in that vector to c_j. So, β sets C_j to true. As this holds for all clauses, we know that $[\![\varphi]\!]_\beta = 1$, concluding this direction of the reduction and the proof overall. $\square$

Example 21. *For comparison, we again consider the same* 3*-CNF-SAT formula as in Example 19, i.e.,*

$$(X_1 \vee X_2 \vee \neg X_3) \wedge (\neg X_1 \vee X_2 \vee X_4).$$

As before, we can also write this instance as

$$\varphi = \{\{X_1, X_2, \neg X_3\}, \{\neg X_1, X_2, X_4\}\}.$$

Hence, we have $m = 4$ distinct variables and $n = 2$ distinct clauses. In contrast to the previous examples, the dimensionality of each data-point is now fixed to $d = 3$. However, we use an unbounded alphabet containing the symbols

$$x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}, x_{3,1}, x_{3,2}, x_{4,1}, x_{4,2}, c_1, c_2, \#.$$

The symbols $x_{i,1}$ and $x_{i,2}$ are used to enforce a consistent choice for the truth value of the variable X_i, while c_1 and c_2 represent the two clauses of the input formula.

Following the construction from Proposition 8, we only construct one input cluster A_1. For each variable X_i, we obtain two circles of length 2*. The first circle corresponds to setting X_i to true, while the second circle corresponds to setting X_i to false. As the two clauses are*

$$C_1 = \{X_1, X_2, \neg X_3\} \quad \textit{and} \quad C_2 = \{\neg X_1, X_2, X_4\},$$

we obtain the following input cluster:

$$\begin{aligned} A_1 = \{&(x_{1,1}, x_{1,1}, c_1), (x_{1,2}, x_{1,2}, \#), (x_{1,1}, x_{1,2}, \#), (x_{1,2}, x_{1,1}, c_2),\\ &(x_{2,1}, x_{2,1}, c_1), (x_{2,2}, x_{2,2}, c_2), (x_{2,1}, x_{2,2}, \#), (x_{2,2}, x_{2,1}, \#),\\ &(x_{3,1}, x_{3,1}, \#), (x_{3,2}, x_{3,2}, \#), (x_{3,1}, x_{3,2}, c_1), (x_{3,2}, x_{3,1}, \#),\\ &(x_{4,1}, x_{4,1}, \#), (x_{4,2}, x_{4,2}, c_2), (x_{4,1}, x_{4,2}, \#), (x_{4,2}, x_{4,1}, \#),\\ &(\#, \#, \#)\}. \end{aligned}$$

Here, for example, the two vectors $(x_{1,1}, x_{1,1}, c_1)$ and $(x_{1,2}, x_{1,2}, \#)$ correspond to setting X_1 to true. This satisfies the first clause but not the second one. On the other hand, the two vectors $(x_{1,1}, x_{1,2}, \#)$ and $(x_{1,2}, x_{1,1}, c_2)$ correspond to setting X_1 to false, which satisfies the second clause but not the first one.

The target cluster I contains, for each variable X_i, the two vectors enforcing that both $x_{i,1}$ and $x_{i,2}$ have to occur in both of the first two coordinates. In addition, it contains one vector for each clause symbol c_1 and c_2. Thus, we obtain

$$\begin{aligned} I = \{&(x_{1,1}, x_{1,1}, \#), (x_{1,2}, x_{1,2}, \#), (x_{2,1}, x_{2,1}, \#), (x_{2,2}, x_{2,2}, \#),\\ &(x_{3,1}, x_{3,1}, \#), (x_{3,2}, x_{3,2}, \#), (x_{4,1}, x_{4,1}, \#), (x_{4,2}, x_{4,2}, \#),\\ &(\#, \#, c_1), (\#, \#, c_2)\}. \end{aligned}$$

Finally, we set $k_{A_1} = m \cdot n + 1 = 9$. This forces us to select exactly two vectors for each variable and additionally the vector $(\#, \#, \#)$. In particular, for each variable, we cannot mix the two circles. For example, choosing $(x_{1,1}, x_{1,1}, c_1)$ and $(x_{1,2}, x_{1,1}, c_2)$ would cover $x_{1,1}$ and $x_{1,2}$ in the first coordinate, but it would not cover $x_{1,2}$ in the second coordinate. Hence, choosing one vector from one circle enforces choosing the remaining vector from the same circle as well.

For this instance, we can select the following subset:

$$\begin{aligned} A_1' = \{&(x_{1,1}, x_{1,1}, c_1), (x_{1,2}, x_{1,2}, \#), (x_{2,1}, x_{2,1}, c_1), (x_{2,2}, x_{2,2}, c_2),\\ &(x_{3,1}, x_{3,1}, \#), (x_{3,2}, x_{3,2}, \#), (x_{4,1}, x_{4,1}, \#), (x_{4,2}, x_{4,2}, c_2),\\ &(\#, \#, \#)\}. \end{aligned}$$

This subset corresponds to setting all variables X_1, X_2, X_3, X_4 to true. It VE-covers all vectors in I: the $x_{i,j}$-symbols are covered in the first and second coordinates by the selected true-circles, the $\#$-symbols are covered by $(\#, \#, \#)$, and the clause symbols c_1 and c_2 occur in the third coordinate of selected vectors. Thus, the constructed instance is indeed a yes-instance, reflecting that the original 3*-CNF-SAT instance is satisfiable.*

A.2.4 Proof and Example for Proposition 10

Proof. The containment in NP follows from Proposition 3. As in the previous results, we obtain NP-hardness by a reduction from 3-CNF-SAT. Let $\varphi = \{C_1, ..., C_n\}$ be a formula in 3-conjunctive normal form with variables $\{X_1, ..., X_m\}$ such that $C_i = \{X_{i,1}, X_{i,2}, X_{i,3}\}$ for literals $X_{i,j} \in \{X_1, \overline{X}_1, ..., X_m, \overline{X}_m\}$. The construction and idea work very similarly to the previous one in Proposition 8. Again, we construct two parallel circles for each variable, each corresponding to setting either the positive or negated one to 1. This time, however, we do not have to compress the construction into one cluster. As we need a constant bound on the maximum number of elements per cluster, we also cannot split up per variable, as the circle length depends on n. Hence, we will utilize $n \cdot m$ many clusters with selection bounds of 1, consisting of only 2 vectors and where each vector corresponds to one element of the circles/subsets introduced in the construction of Proposition 8. Hence, the constants utilized are, again, a dimensionality of 3 per data-point and a total number of 2 elements per cluster. The symbols in Σ are analogous to the ones used in the proof of Proposition 8.

We proceed with the specific construction: For each $i \in [m]$, we construct clusters $A_{i,1}$ to $A_{i,n}$ by setting, for each $j \in [n-1]$, $A_{i,j} = \{(x_{i,j}, x_{i,j}, C_{i,t,j}), (x_{i,j}, x_{i,j+1}, C_{i,f,j})\}$ with the additional bound of $k_{A_{i,j}} = 1$. In the case of $j = n$, we set $A_{i,n} = \{(x_{i,n}, x_{i,n}, C_{i,t,n}), (x_{i,n}, x_{i,1}, C_{i,f,n})\}$ with the same bound $k_{A_{i,n}} = 1$. For each $i \in [m]$, $j \in [n]$, and $b \in \{t, f\}$, we set $C_{i,b,j}$ analogously to the proof of Proposition 8. To obtain the $(\#, \#, \#)$ vector, we construct a final cluster $A_\# = \{(\#, \#, \#)\}$ with $k_{A_\#} = 1$ that allows for the choice of only the $(\#, \#, \#)$ vector. Finally, we also set I analogously to the construction given in 8.

Similar to before, we have to argue that the selected vectors for each variable X_i cannot mix the two circles. Fix some $i \in [m]$. For each $j \in [n]$, the symbol $x_{i,j}$ occurs in the first coordinate only in vectors from the cluster $A_{i,j}$. As the target cluster I contains the vector $(x_{i,j}, x_{i,j}, \#)$, the symbol $x_{i,j}$ has to be covered in the first coordinate. Hence, from each cluster $A_{i,j}$, at least one vector has to be selected. Since the selection bound is $k_{A_{i,j}} = 1$, exactly one vector is selected from each $A_{i,j}$.

Now consider the second coordinates of the selected vectors for this fixed variable X_i. If we select

$$(x_{i,j}, x_{i,j}, C_{i,t,j})$$

from $A_{i,j}$, then we obtain $x_{i,j}$ in the second coordinate. If we select

$$(x_{i,j}, x_{i,j+1}, C_{i,f,j}),$$

where $x_{i,n+1}$ is interpreted as $x_{i,1}$, then we obtain $x_{i,j+1}$ in the second coordinate. Since I contains all vectors $(x_{i,j}, x_{i,j}, \#)$, for $j \in [n]$, all values $x_{i,1}, \ldots, x_{i,n}$ also have to occur in the second coordinate of selected vectors.

Assume that the choices for X_i are mixed. Then, as the indices are considered modulo n, there exists some $j \in [n]$ such that the false-circle vector is selected from $A_{i,j-1}$ while the true-circle vector is selected from $A_{i,j}$. But then both selected vectors contribute the same value $x_{i,j}$ in the second coordinate:

$$(x_{i,j-1}, x_{i,j}, C_{i,f,j-1}) \quad \text{and} \quad (x_{i,j}, x_{i,j}, C_{i,t,j}).$$

As exactly one vector is selected from each of the n clusters $A_{i,1}, \ldots, A_{i,n}$, this repetition implies that at least one value from $\{x_{i,1}, \ldots, x_{i,n}\}$ is missing in the second coordinate. This contradicts the fact that I has to be VE-covered. Therefore, for each variable X_i, the selected vectors are either all from the true-circle

$$\{(x_{i,1}, x_{i,1}, C_{i,t,1}), (x_{i,2}, x_{i,2}, C_{i,t,2}), \ldots, (x_{i,n}, x_{i,n}, C_{i,t,n})\}$$

or all from the false-circle

$$\{(x_{i,1}, x_{i,2}, C_{i,f,1}), (x_{i,2}, x_{i,3}, C_{i,f,2}), \ldots, (x_{i,n}, x_{i,1}, C_{i,f,n})\}.$$

Hence, also in this construction, every VE-covering selection induces a consistent truth value for each variable.

Regarding the computation time of this construction, we observe that we have to construct $m \cdot n$ many clusters, each cluster consisting of a constant number of elements of constant dimensionality. Finally, the target cluster I consists of $mn + n$ many elements of constant dimensionality. As checking for the existence of literals in clauses can also be assumed to have a runtime upper bounded by a constant,

we end up with a computation time of $O((mn+n)+mn) = O(m \cdot n)$. We proceed with the proof of correctness of the reduction.

($\Rightarrow$): Assume there exists a satisfying assignment $\beta : \{X_1, ..., X_m\} \to \{0,1\}$ of φ. For each $i \in [m]$, if $\beta(X_i) = 1$, we select from each $A_{i,j}$, for $j \in [n]$, the subset $A'_{i,j} = \{(x_{i,j}, x_{i,j}, C_{i,t,j})\}$. This satisfies the bound $|A'_{i,j}| = 1 = k_{A_{i,j}}$. Otherwise, if $\beta(X_i) = 0$ (resulting in $[\![\overline{X}_i]\!]_\beta = 1$), we select for each $j \in [n-1]$ the subsets $A'_{i,j} = \{(x_{i,j}, x_{i,j+1}, C_{i,f,j})\}$ and the subset $A'_{i,n} = \{(x_{i,n}, x_{i,1}, C_{i,f,n})\}$. Finally, we also select the $A'_\# = A_\#$ as the final set that only contains the $(\#, \#, \#)$ vector. As observed already in the proof of Proposition 8, this covers all vectors of the form $(x_{i,j}, x_{i,j}, \#)$ in I. For each C_j, with $j \in [n]$, we know that there must exist some literal $X_{j,\ell}$, with $\ell \in \{1,2,3\}$, with $[\![X_{j,\ell}]\!]_\beta = 1$. We know that this literal corresponds to some X_i or $\overline{X}_i$. Hence, there exists a sequence of selected subsets $A'_{i,1}$ to $A'_{i,n}$ corresponding to that literal. By the construction of $A_{i,1}$ to $A_{i,n}$ and the selections $A'_{i,1}$ to $A'_{i,n}$, we have that the selected vector in $A'_{i,j}$ has the value c_j at position 3. As this holds for all clauses C_j, with $j \in [n]$, we know that we also cover all vectors of the form $(\#, \#, c_j)$ in I. Therefore, $A'_\# \cup \bigcup_{i=1}^{m}(\bigcup_{j=1}^{n} A'_{i,j})$ VE-covers I, concluding this direction.

($\Leftarrow$): Now, assume there exist subsets $A'_{1,1}, ..., A'_{1,n}, ..., A'_{m,1}, ..., A'_{m,n}, A'_\#$ of $A_{1,1}, ..., A_{1,n}, ..., A_{m,1}, ..., A_{m,n}, A_\#$, respectively, such that $A' = A'_\# \cup \bigcup_{i=1}^{m}(\bigcup_{j=1}^{n} A'_{i,j})$ VE-covers I. For each $i \in [m]$, by the arguments from the proof of Proposition 8 and the ones mentioned before, we know that the selected vectors in $A'_{i,1}$ to $A'_{i,n}$ all either fall in the set

$$\{(x_{i,1}, x_{i,1}, C_{i,t,1}), (x_{i,2}, x_{i,2}, C_{i,t,2}), ..., (x_{i,n}, x_{i,n}, C_{i,t,n})\}$$

or in the set

$$\{(x_{i,1}, x_{i,2}, C_{i,f,1}), (x_{i,2}, x_{i,3}, C_{i,f,2}), ..., (x_{i,n}, x_{i,1}, C_{i,f,n})\}.$$

We construct an assignment $\beta : \{X_1, ..., X_m\} \to \{0,1\}$ as follows. For each $i \in [m]$, if the selected vectors in $A'_{i,1}$ to $A'_{i,n}$ fall into the first set, we set $\beta(X_i) = 1$. Otherwise, if they fall in the second set, we set $\beta(X_i) = 0$ (resulting in $[\![\overline{X}_i]\!]_\beta = 1$). Now, consider the clause C_j, for some $j \in [n]$. We know that the symbol c_j occurs in some selected vector in A'. Let $i \in [m]$ such that $(x_{i,j}, x_{i,a}, c_j) \in A'$, where $a \in \{j, j+1, 1\}$. Then, by the given constructions, we know that there exists a literal $X_{j,\ell}$, with $\ell \in \{1,2,3\}$ in C_j that corresponds to either X_i or $\overline{X}_i$. By the selection in A' and the construction of β, we know that $[\![X_{j,\ell}]\!]_\beta = 1$. As we can do that for all clauses C_j with $j \in [n]$, we know that $[\![\varphi]\!]_\beta = 1$, hence resulting in satisfiability of φ. This concludes this direction and, therefore, the whole proof. □

Example 22. *We again consider the same 3-CNF-SAT formula as before:*

$$(X_1 \vee X_2 \vee \neg X_3) \wedge (\neg X_1 \vee X_2 \vee X_4).$$

Hence, the formula contains $m = 4$ variables and $n = 2$ clauses. The construction used in Proposition 10 is very similar to the construction from Proposition 8. The main difference is that we no longer collect all vectors in one single cluster. Instead, we split the circles into smaller clusters, each containing only two vectors. This keeps the number of elements per cluster bounded by the constant 2.

The target cluster I is the same as in Example 21, namely

$$\begin{aligned} I = \{&(x_{1,1}, x_{1,1}, \#), (x_{1,2}, x_{1,2}, \#), (x_{2,1}, x_{2,1}, \#), (x_{2,2}, x_{2,2}, \#),\\ &(x_{3,1}, x_{3,1}, \#), (x_{3,2}, x_{3,2}, \#), (x_{4,1}, x_{4,1}, \#), (x_{4,2}, x_{4,2}, \#),\\ &(\#, \#, c_1), (\#, \#, c_2)\}. \end{aligned}$$

For each variable X_i and each clause position $j \in \{1,2\}$, we construct one cluster $A_{i,j}$ with selection bound $k_{A_{i,j}} = 1$. In addition, we construct the cluster $A_\# = \{(\#, \#, \#)\}$ with $k_{A_\#} = 1$. For the concrete formula above, this gives us the following clusters:

$$\begin{aligned} A_{1,1} &= \{(x_{1,1}, x_{1,1}, c_1), (x_{1,1}, x_{1,2}, \#)\}, \quad A_{1,2} = \{(x_{1,2}, x_{1,2}, \#), (x_{1,2}, x_{1,1}, c_2)\},\\ A_{2,1} &= \{(x_{2,1}, x_{2,1}, c_1), (x_{2,1}, x_{2,2}, \#)\}, \quad A_{2,2} = \{(x_{2,2}, x_{2,2}, c_2), (x_{2,2}, x_{2,1}, \#)\},\\ A_{3,1} &= \{(x_{3,1}, x_{3,1}, \#), (x_{3,1}, x_{3,2}, c_1)\}, \quad A_{3,2} = \{(x_{3,2}, x_{3,2}, \#), (x_{3,2}, x_{3,1}, \#)\},\\ A_{4,1} &= \{(x_{4,1}, x_{4,1}, \#), (x_{4,1}, x_{4,2}, \#)\}, \quad A_{4,2} = \{(x_{4,2}, x_{4,2}, c_2), (x_{4,2}, x_{4,1}, \#)\},\\ A_\# &= \{(\#, \#, \#)\}. \end{aligned}$$

In each cluster $A_{i,j}$, the first vector corresponds to the jth vector of the circle that sets X_i to true, while the second vector corresponds to the jth vector of the circle that sets X_i to false. Since the selection bound of each cluster is 1, we have to locally choose between the true-circle and the false-circle. The target vectors then enforce that this choice is globally consistent for each variable. For example, if we select $(x_{1,1}, x_{1,1}, c_1)$ from $A_{1,1}$, then selecting $(x_{1,2}, x_{1,1}, c_2)$ from $A_{1,2}$ would not cover $x_{1,2}$ in the second coordinate. Hence, we are forced to also select $(x_{1,2}, x_{1,2}, \#)$ from $A_{1,2}$.

Again, choosing the true-circle for every variable gives us a VE-covering. More precisely, we can select

$$\begin{aligned} A'_{1,1} &= \{(x_{1,1}, x_{1,1}, c_1)\}, \quad A'_{1,2} = \{(x_{1,2}, x_{1,2}, \#)\}, \\ A'_{2,1} &= \{(x_{2,1}, x_{2,1}, c_1)\}, \quad A'_{2,2} = \{(x_{2,2}, x_{2,2}, c_2)\}, \\ A'_{3,1} &= \{(x_{3,1}, x_{3,1}, \#)\}, \quad A'_{3,2} = \{(x_{3,2}, x_{3,2}, \#)\}, \\ A'_{4,1} &= \{(x_{4,1}, x_{4,1}, \#)\}, \quad A'_{4,2} = \{(x_{4,2}, x_{4,2}, c_2)\}, \\ A'_{\#} &= \{(\#, \#, \#)\}. \end{aligned}$$

This selection satisfies all selection bounds and corresponds to the assignment that maps all variables to true. The selected vectors cover all symbols $x_{i,j}$ in the first and second coordinates, the vector $(\#, \#, \#)$ covers all required occurrences of $\#$, and the symbols c_1 and c_2 occur in the third coordinate of selected vectors. Hence, the union of all selected subsets VE-covers I, which again reflects the satisfiability of the original 3-CNF-SAT instance.

A.3 P-Containment Results

A.3.1 Proof of Proposition 12

Proof. Assume $|\Sigma|$ to be upper bounded by the constant $s \in \mathbb{N}$ and assume $d \in \mathbb{N}$ to be the constant upper bound for the dimensionality of the considered data points. We assume the following input: Let $A_1, ..., A_n \subseteq \Sigma^d$ be n many clusters of w.l.o.g. d-dimensional data-points. Let $k_{A_1}, ..., k_{A_n} \in \mathbb{N}$ be some numbers representing the selection bounds for each cluster respectively. Assume w.l.o.g. $k_{A_i} \leq |A_i|$, for all $i \in [n]$. Finally, let $I \subseteq \Sigma^d$ be some target cluster. Using the provided constants, we can construct a polynomial time algorithm to decide P-`Inbetween-Decision`.

Note, given $|\Sigma| \leq s$ and a dimensionality at most d, that we can construct at most $V = s^d$ many distinct vectors. As s and d are constants, clearly, V is also a constant. Hence, $|A_i| \leq V$, for all $i \in [n]$, and also $|I| \leq V$. In each of the at most d coordinates, there can occur at most s distinct values that have to be covered in I. Hence, in total there are at most $s \cdot d$ many distinct values to cover in I. Therefore, there are $C = 2^{s \cdot d}$ many states representing partial coverings of I. Additionally, we know that, for each $i \in [n]$ there are at most

$$K = 2^V$$

many distinct possibilities to pick at most k_{A_i} many elements from the ith cluster. As V is upper bounded by a constant, k_{A_i} is also upper bounded by a constant, which in turn also leads to K being upper bounded by a constant independent from the other, potentially unbounded, input parameters (e.g., the number of clusters).

We continue with the following procedure to decide P-`Inbetween-Decision`. Let $Cov = \{\emptyset\}$ be a set initialized with the empty set as its element. Now, for each cluster A_i, with $i \in [n]$, we do the following: For all currently reached partial coverings $q \in Cov$, and for all possible subsets $A'_i \subseteq A_i$ with $|A'_i| \leq k_{A_i}$, we compute all additional elements covered in I by adding the elements from A'_i to the elements already covered by q. Let q' be the new set returned by this. We add q' to a new set Cov'. Finally, once we have iterated through all elements in Cov and checked for each such element all A'_i to compute elements for Cov', we replace Cov by Cov' and continue with the next cluster. Once we have performed that procedure for all clusters, we check whether there exists an element $q \in Cov$ that is a full covering of the elements in I. If such an element q exists, we return true, otherwise we return false.

(Completeness) First, assume there exist subsets $A'_1, ..., A'_n$ with $|A'_i| \leq k_{A_i}$, for all $i \in [n]$ such that $\bigcup_{i \in [n]} A'_i$ VE-covers I. In the first iteration, the algorithm will test A'_1 and then add a set representing the elements covered by A'_1 to Cov. Now assume that the algorithm does so successfully up to

some $i \in [n-1]$. Then, we can assume there exists an element $q \in Cov$ that represents exactly the elements covered in I by $\bigcup_{j \in [i]} A'_j$. Now, the algorithm will consider q and also compute the extended partial covering of I we obtain by adding the subset A'_{i+1}. Doing so inductively up to n will result in the algorithm returning true.

(Correctness) Now, for the other direction, assume the algorithm returns true. Then there exists a sequence of partial coverings $(q_0, q_1, ..., q_n)$, with $q_0 = \emptyset$ and q_n representing all elements from I being covered, such that, for each $i \in [n]$, there exists a subset $A'_i \subseteq A_i$ with $|A'_i| \leq k_{A_i}$ for which we have that the partial covering represented by q_i represents the partial covering q_{i-1} plus the additional elements obtained by adding A'_i. As we start with the empty set, by induction, we obtain that $\bigcup_{i \in [n]} A'_i$ VE-covers I.

(Runtime) The outer loop iterates through n many clusters. For each cluster, we know that $|Cov| \leq C$ is upper bounded by a constant. Also, for each $i \in [n]$ and for each $q \in Cov$, there are at most K distinct subsets $A'_i \subseteq A_i$ to check. Computing the extended covering q' from q of I based on $|A'_i|$ can be done by checking, for each element $p \in A'_i$ and each coordinate $\ell \in [d]$, whether $p[\ell]$ occurs in the set of elements to be covered in coordinate ℓ of I. As $|A'_i|$ is upper bounded by V as well as $|I|$, we obtain a total runtime of $\mathcal{O}(n \cdot C \cdot K \cdot V^2 \cdot d)$. As C, K, and V are independent from n and only based on s and d, there exists a function $f : \mathbb{N}^2 \to \mathbb{R}_{\geq 0}$ such that $f(s,d) = C \cdot K \cdot V^2 \cdot d$. Hence, the total runtime of the algorithm can also be described as $\mathcal{O}(f(s,d) \cdot n)$. In particular, `Inbetween-Decision` admits a fixed-parameter linear-time algorithm with respect to the alphabet size together with the maximum dimensionality of data-points. □

A.3.2 Proof of Proposition 13

Proof. Assume the total number of input clusters to be bounded by a constant $n \in \mathbb{N}$ and assume the number of elements per cluster to be bounded by a constant $m \in \mathbb{N}$. We assume the following input: Let Σ be some set of symbols and assume $|\Sigma| = s$. Let $d \in \mathbb{N}$ be the dimensionality of data-points considered. Let $A_1, ..., A_{n'} \subseteq \Sigma^d$ be some n' many input clusters where $n' \leq n$. For each input cluster A_i, with $i \in [n']$, assume $|A_i| \leq m$. Finally, let $I \subseteq \Sigma^d$ be the target cluster to be checked. Using the provided constants, we can construct a polynomial time algorithm that decides P-`Inbetween-Decision`.

By the bound m on the number of elements per input cluster A_i, for $i \in [n']$, we may assume w.l.o.g. $k_{A_i} \leq m$ as well. Therefore, there are at most $K = 2^m$ distinct subsets A'_i that can be picked from A_i with $|A'_i| \leq k_{A_i} \leq m$. As the number of clusters is also bounded by n, we know that there are at most K^n distinct ways to pick subsets $A'_1, ..., A'_{n'}$ of A_1 to $A_{n'}$, respectively. Hence, we can just do the following: For each distinct selection of subsets $A'_1 \subseteq A_1$ to $A'_{n'} \subseteq A_{n'}$, with $|A'_i| \leq k_{A_i}$, respectively, check whether $\bigcup_{i \in [n']} A'_i$ VE-covers I. To check each VE-covering, we can just map all $|I| \cdot d$ distinct elements to cover in I onto a $|I| \cdot d$ matrix, iterate through A'_1 to $A'_{n'}$ and for each element of a given dimension $\ell \in [d]$, we check the existence of that element in the $|I|$ many elements occurring in dimension ℓ in I. If we hit an element, we place a marker in the respective location in the constructed matrix. If at some point all fields in the constructed matrix have a marker, we return true as we found a VE-covering of I in the given bounds. Otherwise, if no selection of subsets suffices, we return false. The correctness and completeness of this approach follow by the brute force nature of it.

(Runtime) We have already identified the upper bound $K^n = (2^m)^n$ for the number of distinct sets of subsets $A'_1, ..., A'_{n'}$ of A_1 to $A_{n'}$ that we have to check. For each such set of subsets, checking whether it is a VE-covering of I takes at most $\mathcal{O}(d \cdot \sum_{i=1}^{n'} (|A'_i|) \cdot |I|)$ time, which lies in $\mathcal{O}(dnm \cdot |I|)$. Hence, overall, we obtain a runtime of $\mathcal{O}(K^n \cdot dnm|I|)$. As K^n is bounded by $O(1)$, it is contained $\mathcal{O}(dnm|I|)$. Overall, we obtain a runtime of

$$\mathcal{O}\left(2^{mn} \cdot dnm|I|\right).$$

Let N denote the size of the input. Clearly, both d and $|I|$ are upper bounded by N. Furthermore, the factors n and m depend only on the two considered parameters. Hence, by defining

$$f(n,m) = 2^{mn} \cdot n \cdot m,$$

the runtime is upper bounded by

$$\mathcal{O}\left(f(n,m) \cdot N^2\right).$$

In particular, Inbetween-Decision is fixed-parameter tractable with respect to the number of input clusters together with the maximum number of elements per input cluster. □

A.3.3 Proof of Proposition 14

Proof. Let $d \in \mathbb{N}$ be the constant maximum dimensionality of data-points considered in this case. Let $t \in \mathbb{N}$ be the constant upper bound for $|I|$. We assume the following input: Let Σ be some set of symbols and assume $|\Sigma| = s$. Let $A_1, ..., A_n \subseteq \Sigma^d$ be some n many input clusters with w.l.o.g. d-dimensional data-points. Let

$$m = \max_{i \in [n]}(|A_i|)$$

be the size of the cluster with the most elements. Finally, let $I \subseteq \Sigma^d$ be the target cluster to be checked. Using the provided constants, we can construct a polynomial time algorithm that decides P-Inbetween-Decision.

Due to the bounds d and t, we know that there are at most $d \cdot t$ distinct elements to cover in I, with each of the d dimensions having at most t distinct elements to cover. That results in $C = 2^{d \cdot t}$ distinct partial coverings of elements in I that can be represented by some state q encoding the partial covering in $\mathcal{O}(d \cdot t)$ space. Therefore, I determines the only relevant elements from each input cluster A_i, $i \in [n]$. For each dimension $\ell \in [d]$, if there exist letters $a, b \in \Sigma$ that do not occur in any vector from I at position ℓ, neither character contributes anything to the partial solutions, hence we could assume a and b to be equal, resulting in only one additional character $\# \notin \Sigma$ that can be used to represent all characters not occurring in I. For each coordinate, at most t symbols occurring in I are relevant, together with the dummy symbol $\#$. Hence, there are at most $V = (t+1)^d$ distinct relevant projections. For each A_i, we can compute the relevant projection of every vector and keep only one representative of each distinct projected vector. Since there are at most $V = (t+1)^d$ distinct projected vectors, every projection only has to be compared with at most V already obtained representatives. Hence, B_i can be constructed in

$$\mathcal{O}\left(f(d,t) \cdot |A_i|\right)$$

time for some computable function f, and we obtain $|B_i| \leq V$.

Using this information, we can proceed with a similar algorithm to the one proposed in Proposition 12.

First, determine the set of characters $\Sigma' \subseteq \Sigma$ that occur in I as some elements. Introduce a new character $\# \notin \Sigma'$. Now, for each dimension $\ell \in [d]$, determine the characters occurring in dimension ℓ of the elements in I and store them in a set D_ℓ, in particular,

$$D_\ell = \{\ a \in \Sigma \mid \exists p \in I : p[\ell] = a\ \}.$$

Next, for each $i \in [n]$, filter A_i by relevant vectors and remove duplicates with respect to elements relevant for VE-coverings of I. In particular, for each vector $p \in A_i$, define its relevant projection $\rho(p)$ by setting, for each $\ell \in [d]$,

$$\rho(p)[\ell] = \begin{cases} p[\ell] & \text{if } p[\ell] \in D_\ell, \\ \# & \text{otherwise.} \end{cases}$$

Now, for each $i \in [n]$, we define

$$B_i = \{\rho(p) \mid p \in A_i\}.$$

Notice that B_i is not necessarily a subset of A_i, but a set of representatives of the relevant behaviour of vectors in A_i with respect to VE-coverings of I. Since, for each coordinate $\ell \in [d]$, there are at most t relevant symbols and one additional dummy symbol $\#$, there are at most $(t+1)^d$ many distinct vectors in B_i. Hence, $|B_i| \leq V$ for $V = (t+1)^d$. As V is bounded by a constant, we only have to consider at most

$$K = 2^V$$

many relevant subsets of B_i for each cluster A_i. This is again bounded by a constant depending only on d and t.

We proceed with the following: Introduce a set $Cov = \{\emptyset\}$ that initially only contains the empty-set. The set Cov will be used to represent the partial coverings of I obtained by trying out different relevant subsets of the sets B_i. Now, for each cluster A_i, where $i \in [n]$, and for each $q \in Cov$, we try

all distinct combinations of picking subsets $B'_i \subseteq B_i$ of size at most k_{A_i}. For each such subset B'_i we compute, based on q, the new extended partial covering obtained by adding the elements from B'_i. By that, we obtain a new partial covering q' that we will store, together with all new partial coverings obtained in this iteration (hence for this current cluster), in a new set Cov'. Once we have done that for all $q \in Cov$ and for all possible subsets $B'_i \subseteq B_i$ with $|B'_i| \leq k_{A_i}$, we replace the contents of Cov with the ones from Cov', empty out Cov' for the next iteration, and continue with the next cluster. In the end, if there exists a partial covering $q \in Cov$ that actually covers all elements in I dimension-wise, we return true. Otherwise, we return false.

(Completeness) First, assume there exist subsets $A'_1, ..., A'_n$ with $A'_i \subseteq A_i$ and $|A'_i| \leq k_{A_i}$, for all $i \in [n]$, such that $\bigcup_{i\in[n]} A'_i$ VE-covers I. We show that the algorithm returns true. For each $i \in [n]$, let

$$B'_i = \{\rho(p) \mid p \in A'_i\} \subseteq B_i.$$

Clearly, we have $|B'_i| \leq |A'_i| \leq k_{A_i}$. Furthermore, replacing a vector p by $\rho(p)$ does not change which relevant symbols from I are covered. Indeed, if $p[\ell] \in D_\ell$, then $\rho(p)[\ell] = p[\ell]$, and if $p[\ell] \notin D_\ell$, then $p[\ell]$ does not contribute to covering I in coordinate ℓ anyway. Hence,

$$\bigcup_{i\in[n]} B'_i$$

covers exactly the same relevant coordinate-values of I as $\bigcup_{i\in[n]} A'_i$.

Now, in the first iteration, the algorithm tries the subset B'_1 and therefore stores the partial covering obtained by B'_1 in Cov. Assume inductively that, after considering the clusters $A_1, ..., A_i$, there exists some state $q \in Cov$ representing the partial covering obtained by $\bigcup_{j\in[i]} B'_j$. In the next iteration, the algorithm also tries the subset $B'_{i+1} \subseteq B_{i+1}$ and therefore computes the partial covering obtained by adding B'_{i+1} to q. Thus, by induction, after processing all clusters, the set Cov contains a state representing the covering obtained by $\bigcup_{i\in[n]} B'_i$. As this union VE-covers I, the algorithm returns true.

(Correctness) For the other direction, assume the algorithm returns true. Then, after processing all clusters, there exists a state $q \in Cov$ representing a full VE-covering of I. By the construction of the algorithm, this state was obtained by choosing, for each $i \in [n]$, some subset $B'_i \subseteq B_i$ with $|B'_i| \leq k_{A_i}$ such that $\bigcup_{i\in[n]} B'_i$ covers all relevant coordinate-values of I.

It remains to translate these selected relevant vectors back to vectors from the original clusters. By the definition of B_i, for each vector $b \in B'_i$ there exists some original vector $p_b \in A_i$ such that $\rho(p_b) = b$. Let

$$A'_i = \{p_b \mid b \in B'_i\}.$$

Then $A'_i \subseteq A_i$ and $|A'_i| \leq |B'_i| \leq k_{A_i}$. Moreover, for every coordinate $\ell \in [d]$ and every relevant symbol $a \in D_\ell$, if $b[\ell] = a$, then the chosen representative p_b also satisfies $p_b[\ell] = a$. Thus, every relevant coordinate-value of I that is covered by $\bigcup_{i\in[n]} B'_i$ is also covered by $\bigcup_{i\in[n]} A'_i$. Since the former is a full VE-covering of I, the latter is also a full VE-covering of I. Hence, the original instance is a yes-instance.

(Runtime) We first determine the sets D_ℓ, for all $\ell \in [d]$, by iterating over the at most t vectors in I and their d coordinates. This takes $\mathcal{O}(d \cdot t)$ time. Afterwards, for each input vector $p \in A_i$, we compute the relevant projection $\rho(p)$ by checking, for each coordinate $\ell \in [d]$, whether $p[\ell] \in D_\ell$. This can be done in $\mathcal{O}(d \cdot t)$ time per vector by a direct lookup through the at most t elements of D_ℓ. To obtain B_i, we additionally have to keep only distinct projected vectors. Since there are at most $V = (t + 1)^d$ distinct projections, each newly computed projection has to be compared with at most V already stored vectors. Comparing two projected vectors takes $O(d)$ time. Hence, constructing all sets B_i, for $i \in [n]$, takes time

$$\mathcal{O}\left((d \cdot t + V \cdot d) \cdot \sum_{i=1}^{n} |A_i|\right).$$

Since V, d, and t depend only on the two considered parameters, this part is linear in the number of input vectors up to a factor depending only on d and t.

Afterwards, the dynamic programming procedure processes all n clusters. For each cluster, there are at most

$$C = 2^{d \cdot t}$$

many possible partial covering states. Furthermore, for each cluster A_i, there are at most

$$K \leq 2^V$$

many relevant subsets of B_i to try, where $V = (t+1)^d$. For each such subset, computing the resulting extended partial covering can be done by checking at most V selected relevant vectors, each of dimension d, against the at most t relevant values in the corresponding coordinate. Hence, one transition can be computed in time $\mathcal{O}(V \cdot d \cdot t)$. Therefore, the dynamic programming part runs in

$$\mathcal{O}(n \cdot C \cdot K \cdot V \cdot d \cdot t).$$

Combining both parts, we obtain a total runtime of

$$\mathcal{O}\left((d \cdot t + V \cdot d) \cdot \sum_{i=1}^{n} |A_i| + n \cdot C \cdot K \cdot V \cdot d \cdot t\right).$$

As C, K, V, d, and t depend only on the two bounded parameters δ and τ, there exists a computable function $f : \mathbb{N}^2 \to \mathbb{R}_{\geq 0}$ such that the runtime is bounded by

$$\mathcal{O}\left(f(d,t) \cdot \left(n + \sum_{i=1}^{n} |A_i|\right)\right).$$

Thus, the problem is decidable in polynomial time. In particular, `Inbetween-Decision` admits a fixed-parameter linear-time algorithm with respect to the dimensionality of the data-points together with the size of the target cluster. □