

542 A Proofs of algorithm correctness theorems

543 Here we provide proofs for Theorems 4.1 and 5.1, completing the sketches from the main paper.

544 **Theorem 4.1** (Algorithm 4.1 correctness). *Given $w \in \mathcal{W}_h$, compute $w' = \text{COMPRESS}(w) \in \mathcal{W}_r$.*
 545 *(i) $f_{w'} = f_w$, and (ii) w' is incompressible.*

546 *Proof.* (i): Following the steps of the algorithm we rearrange the summation defining f_w to have the
 547 form of $f_{w'}$. For each $x \in \mathbb{R}^n$,

$$\begin{aligned}
 f_w(x) &= d + \sum_{i=1}^h a_i \tanh(b_i x + c_i) \\
 &= d + \sum_{i \notin I} a_i \tanh(c_i) + \sum_{i \in I} a_i \tanh(b_i x + c_i) && \text{(cf. line 3)} \\
 &= \delta + \sum_{i \in I} a_i \tanh(b_i x + c_i) && \text{(cf. line 4)} \\
 &= \delta + \sum_{j=1}^J \sum_{i \in \Pi_j} a_i \tanh(b_i x + c_i) && \text{(cf. line 6)} \\
 &= \delta + \sum_{j=1}^J \sum_{i \in \Pi_j} \text{sign}(b_i) \cdot a_i \tanh(\text{sign}(b_i) \cdot b_i x + \text{sign}(b_i) \cdot c_i) && \text{(tanh odd)} \\
 &= \delta + \sum_{j=1}^J \left(\sum_{i \in \Pi_j} \text{sign}(b_i) \cdot a_i \right) \tanh(\beta_j x + \gamma_j) && \text{(cf. lines 6, 9)} \\
 &= \delta + \sum_{j=1}^J \alpha_j \tanh(\beta_j x + \gamma_j) && \text{(cf. line 8)} \\
 &= \delta + \sum_{j=1}^r \alpha_{k_j} \tanh(\beta_{k_j} x + \gamma_{k_j}) && \text{(cf. line 12)} \\
 &= f_{w'}(x). && \text{(cf. line 14)}
 \end{aligned}$$

548 (ii): Each of the reducibility conditions fails to hold for $w' \in \mathcal{W}_r$: (i) no α_k is zero, due to line 12;
 549 (ii) no β_k is zero, due chiefly to line 3; (iii), (iv) all $\pm(\beta_k, \gamma_k)$ are distinct, due chiefly to line 6. \square

550 **Theorem 5.1** (Algorithm 5.1 correctness). *For $w \in \mathcal{W}_h$ and $\varepsilon \in \mathbb{R}^+$, $\text{prank}_\varepsilon(w) \leq \text{BOUND}(\varepsilon, w)$.*

551 *Proof.* Trace the algorithm to construct a parameter $u \in \bar{B}_\infty(w; \varepsilon)$ with $\text{rank}(u) = \text{BOUND}(\varepsilon, w)$.

552 Construct $u = (a_1^{(u)}, b_1^{(u)}, c_1^{(u)}, \dots, a_h^{(u)}, b_h^{(u)}, c_h^{(u)}, d) \in \mathcal{W}_h$ as follows.

- 553 1. For $i \notin I$, $\|b_i\|_\infty \leq \varepsilon$, so set $b_i^{(u)} = 0$, leaving $a_i^{(u)} = a_i$ and $c_i^{(u)} = c_i$.
- 554 2. For $i \in \Pi_j$, note that $\|\text{sign}(b_i) \cdot (b_i, c_i) - v_j\|_\infty \leq \varepsilon$, so set $(b_i^{(u)}, c_i^{(u)}) = \text{sign}(b_i) \cdot v_j$.
- 555 3. For $i \in \Pi_j$, if $\|\alpha_j\|_\infty \leq \varepsilon \cdot |\Pi_j|$, then set $a_i^{(u)} = a_i - \text{sign}(b_i) \cdot \frac{\alpha_j}{|\Pi_j|}$, else set $a_i^{(u)} = a_i$.

556 By construction, $u \in \bar{B}_\infty(w; \varepsilon)$. To see that $\text{rank}(u) = \text{BOUND}(\varepsilon, w)$, run Algorithm 4.2 on
 557 u : Stage 1 finds the same I , since those $b_i^{(u)} = 0$. Stage 2 finds the same Π_1, \dots, Π_J , since
 558 $\text{sign}(b_i^{(u)}) \cdot (b_i^{(u)}, c_i^{(u)}) = \text{sign}(b_i) \cdot (b_i^{(u)}, c_i^{(u)}) = v_j$ ($\text{sign}(b_i^{(u)}) = \text{sign}(b_i)$ since the first component
 559 of v_j is positive, by line 5 of Algorithm 5.1). Finally, Stage 3 excludes the same α_j , since for j such
 560 that $\|\alpha_j\|_\infty \leq \varepsilon \cdot |\Pi_j|$, these units from u merge into one unit with outgoing weight

$$\sum_{i \in \Pi_j} \text{sign}(b_i^{(u)}) \cdot a_i^{(u)} = \sum_{i \in \Pi_j} \text{sign}(b_i) \cdot \left(a_i - \text{sign}(b_i) \cdot \frac{\alpha_j}{|\Pi_j|} \right) = \sum_{i \in \Pi_j} \text{sign}(b_i) \cdot a_i - \alpha_j = 0. \quad \square$$

B Three perspectives on uniform point cover, and related problems

We show that Problem [UPC](#) is one of three equivalent perspectives on the same abstract decision problem. Each of the three perspectives suggests distinct connections to existing problems. Figure 3 shows example instances of the three problems.

Perspective 1: Uniform point cover. Given some points in the plane, is there a small number of new “covering” points so that each (original) point is near a covering point? This is the perspective introduced in the main paper as Problem [UPC](#). It emphasises the existence of covering points, which are useful in the reduction to Problem [PR](#) for the proof of Theorem 6.2.

Uniform point cover is reminiscent of well-known hard clustering problems such as *Planar k -means* ([Mahajan et al., 2012](#)). The k -means problem concerns finding k centroids with a low *sum* of (squared Euclidean) distances between source points and their nearest centroids. In contrast, uniform point cover concerns finding covering points with a low *maximum* (uniform) distance between source points and their nearest covering points.

Problem [UPC](#) is more similar to the \mathcal{NP} -complete problem of *absolute vertex p -centre* ([Hakimi, 1964, 1965](#); [Kariv and Hakimi, 1979](#)). This problem concerns finding a set of points on a graph (i.e., vertices or points along edges) with a low maximum (shortest path) distance between vertices and their nearest points in the set. Problem [UPC](#) is a geometric p -center problem using uniform distances.

A geometric p -center problem using Euclidean distances was shown to be \mathcal{NP} -complete by [Supowit \(1981, §4.3.2\)](#); see also [Megiddo and Supowit, 1984](#)). [Megiddo and Supowit \(1984\)](#) also showed the \mathcal{NP} -hardness of an optimisation variant using L_1 distance (rectilinear or Manhattan distance).⁹ We prove that Problem [UPC](#) is \mathcal{NP} -complete using somewhat similar reductions, but many simplifications afforded by starting from a more restricted variant of Boolean satisfiability.

Perspective 2: Uniform point partition. Given some points in the plane, can the points be partitioned into a small number of groups with small uniform diameter? This perspective, formalised as Problem [UPP](#), emphasises the grouping of points, rather than the specific choice of covering points.

Consider h points $x_1, \dots, x_h \in \mathbb{R}^2$. A (r, ε) -partition of the h points is a partition of $\{1, \dots, h\}$ into r subsets Π_1, \dots, Π_r such that the uniform distance between points in any subset is at most ε : $\forall k \in \{1, \dots, r\}, \forall i, j \in \Pi_k, \|x_i - x_j\|_\infty \leq \varepsilon$.

Problem UPP. Uniform point partition, or UPP, is a decision problem. Each instance comprises a collection of $h \in \mathbb{N}$ points $x_1, \dots, x_h \in \mathbb{R}^2$, a uniform diameter $\varepsilon \in \mathbb{R}^+$, and a number of groups $r \in \mathbb{N}$. The affirmative instances are those for which there exists an (r, ε) -partition of x_1, \dots, x_h .

Perspective 3: Clique partition for unit square graphs. Given a special kind of graph called a *unit square graph*, can the vertices be partitioned into a small number of cliques? The third perspective strays from the simple neural network context, but reveals further related work.

Consider h points in the plane, $x_1, \dots, x_h \in \mathbb{R}^2$, and a diameter $\varepsilon \in \mathbb{R}^+$. Thus define an undirected graph (V, E) with vertices $V = \{1, \dots, h\}$ and edges $E = \{\{i, j\} : i \neq j, \|x_i - x_j\|_\infty \leq \varepsilon\}$. A *unit square graph*¹⁰ is any graph that can be constructed in this way. Unit square graphs are a uniform-distance variant of unit disk graphs (based on Euclidean distance; cf. [Clark et al., 1990](#)).

Consider an undirected graph (V, E) . A *clique partition* of size r is a partition of the vertices V into r subsets Π_1, \dots, Π_r such that each subset is a clique: $\forall k \in \{1, \dots, r\}, \forall v_i \neq v_j \in \Pi_k, \{v_i, v_j\} \in E$.

Problem usgCP. Clique partition for unit square graphs, or usgCP, is a decision problem. Each instance comprises a unit square graph (V, E) and a number of cliques $r \in \mathbb{N}$. The affirmative instances are those for which there exists a clique partition of size r .

The clique partition problem is \mathcal{NP} -complete in general graphs ([Karp, 1972](#)). [Cerioli et al. \(2004, 2011\)](#) showed that it remains \mathcal{NP} -complete when restricted to unit disk graphs, using a reduction from a variant of Boolean satisfiability that is somewhat similar to our reduction.

⁹The L_1 -distance variant is equivalent to Problem [UPC](#) by a 45° rotation of the plane.

¹⁰“Unit square graph” comes from an equivalent definition of these graphs as intersection graphs of unit squares. To see the equivalence, scale the collection of squares by ε and then consider their centres. The same idea relates the *proximity* and *intersection* models for unit disk graphs ([Clark et al., 1990](#)).

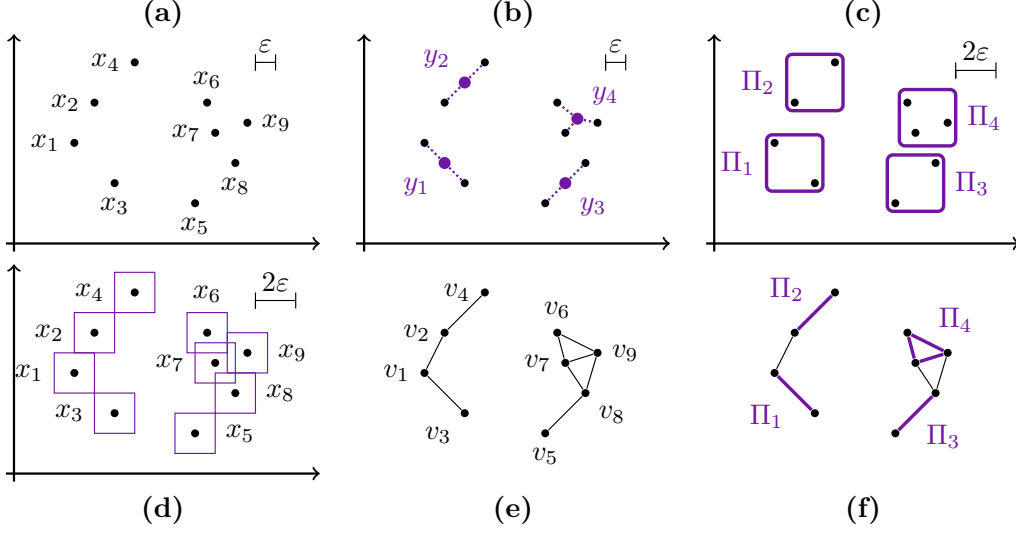


Figure 3: Example instances for Problem **UPC**, Problem **UPP**, and Problem **usgCP**. **(a)** Nine (source) points x_1, \dots, x_9 . **(b)** A $(4, \varepsilon)$ -cover y_1, \dots, y_4 . **(c)** A $(4, 2\varepsilon)$ -partition Π_1, \dots, Π_4 ($= \{1, 3\}, \{2, 4\}, \{5, 8\}, \{6, 7, 9\}$). **(d)** The nine points x_1, \dots, x_9 , along with 2ε -width squares. **(e)** The corresponding unit square graph with vertices v_1, \dots, v_9 . **(f)** A partition of the unit square graph into four cliques Π_1, \dots, Π_4 . **Note:** ε represents a radius in Problem **UPC**, but a diameter in Problems **UPP** and **usgCP**. In these examples, we use ε for the radius and 2ε for the diameter.

Equivalence of the three perspectives. Problems **UPC**, **UPP**, and **usgCP** are equivalent in the sense that there is an immediate reduction between any pair of them.

Theorem B.1 (Equivalence of Problems **UPC**, **UPP**, and **usgCP**). *Let $h, r \in \mathbb{N}$, $\varepsilon \in \mathbb{R}^+$, and $x_1, \dots, x_h \in \mathbb{R}^2$. The following conditions are equivalent:*

- (i) *there exists an $(r, \frac{1}{2}\varepsilon)$ -cover of x_1, \dots, x_h ;*
- (ii) *there exists an (r, ε) -partition of x_1, \dots, x_h ; and*
- (iii) *the unit square graph on x_1, \dots, x_h (diameter ε) has a clique partition of size r .*

Proof. (ii \Rightarrow i): Let Π_1, \dots, Π_r be an (r, ε) -partition of the points x_1, \dots, x_h . For each Π_j , define $y_j \in \mathbb{R}^2$ as the centroid of the bounding rectangle of the set of points $\{x_i : i \in \Pi_j\}$, that is, $y_j = \frac{1}{2}(\max_{i \in \Pi_j} x_{i,1} + \min_{i \in \Pi_j} x_{i,1}, \max_{i \in \Pi_j} x_{i,2} + \min_{i \in \Pi_j} x_{i,2})$. For each $i \in \Pi_j$ and $p \in \{1, 2\}$, let $\alpha = \operatorname{argmax}_{a \in \Pi_j} x_{a,p}$ and $\beta = \operatorname{argmin}_{b \in \Pi_j} x_{b,p}$. Then,

$$\begin{aligned}
 \operatorname{abs}(2x_{i,p} - 2y_{j,p}) &= \operatorname{abs}\left(2x_{i,p} - \left(\max_{i \in \Pi_j} x_{i,p} + \min_{i \in \Pi_j} x_{i,p}\right)\right) \\
 &\leq \operatorname{abs}\left(x_{i,p} - \max_{i \in \Pi_j} x_{i,p}\right) + \operatorname{abs}\left(x_{i,p} - \min_{i \in \Pi_j} x_{i,p}\right) \quad (\text{triangle inequality}) \\
 &= \max_{i \in \Pi_j} x_{i,p} - \min_{i \in \Pi_j} x_{i,p} \quad (\max_{i \in \Pi_j} x_{i,p} \leq x_{i,p} \leq \min_{i \in \Pi_j} x_{i,p}) \\
 &= x_{\alpha,p} - x_{\beta,p} \leq \|x_\alpha - x_\beta\|_\infty \leq \varepsilon.
 \end{aligned}$$

Thus $\|x_i - y_p\|_\infty \leq \frac{1}{2}\varepsilon$, and y_1, \dots, y_k is an $(r, \frac{1}{2}\varepsilon)$ -cover of x_1, \dots, x_h .

(i \Rightarrow iii): Let $y_1, \dots, y_r \in \mathbb{R}^2$ be an $(r, \frac{1}{2}\varepsilon)$ -cover of x_1, \dots, x_h . Partition $\{1, \dots, h\}$ into Π_1, \dots, Π_r by grouping points according to the nearest covering point (break ties arbitrarily). Then for $i, j \in \Pi_k$, $\{i, j\} \in E$ of the unit square graph, since $\|x_i - x_j\|_\infty \leq \|x_i - y_k\|_\infty + \|y_k - x_j\|_\infty \leq \frac{1}{2}\varepsilon + \frac{1}{2}\varepsilon = \varepsilon$. Thus Π_1, \dots, Π_r is a clique partition.

(iii \Rightarrow ii): Let Π_1, \dots, Π_r be a clique partition. Then for $i, j \in \Pi_k$, $\{i, j\} \in E$, and so $\|x_i - x_j\|_\infty \leq \varepsilon$. Thus Π_1, \dots, Π_r is an (r, ε) -partition. \square

625 C \mathcal{NP} -completeness of uniform point cover

626 In this section, we prove that Problem **UPC** is \mathcal{NP} -complete. By Theorem B.1, it suffices to prove
 627 that Problem **UPP** is \mathcal{NP} -complete. This simplifies the presentation of the proof by abstracting
 628 away the need to construct specific covering vectors for groups of points. The main part of the
 629 proof is a reduction from a restricted variant of Boolean satisfiability, which we call Problem **xSAT**.
 630 Appendix C.1 introduces Problem **xSAT** and proves that it is \mathcal{NP} -complete. Appendix C.2 proves
 631 that Problem **UPP** is \mathcal{NP} -complete.

632 C.1 Restricted Boolean satisfiability

633 Boolean satisfiability is a well-known \mathcal{NP} -complete decision problem (Cook, 1971; Levin, 1973).
 634 We formalise this problem as follows.

635 Given n variables, v_1, \dots, v_n , a *Boolean formula (in conjunctive normal form)* is conjunction of m
 636 *clauses*, $c_1 \wedge \dots \wedge c_m$, where each clause is a finite disjunction (\vee) of *literals*, and each literal is either
 637 a variable v_i or its negation \bar{v}_i (called, respectively, a *positive occurrence* or *negative occurrence*
 638 of the variable v_i). A *truth assignment* is a mapping assigning each of the variables v_1, \dots, v_n to the
 639 values “true” or “false.” The formula is *satisfiable* if there exists a truth assignment such that the
 640 entire formula evaluates to “true”. That is, each clause contains at least one positive occurrence of a
 641 variable assigned “true,” or at least one negative occurrence of a variable assigned “false”.

642 **Problem SAT.** Boolean satisfiability, or SAT, is a decision problem. The instances are all Boolean
 643 formulas in conjunctive normal form. The affirmative instances are all satisfiable formulas.

644 We introduce a variant of Problem **SAT**, namely Problem **xSAT**. Let ϕ be a Boolean formulas with
 645 variables v_1, \dots, v_n and clauses $c_1 \wedge \dots \wedge c_m$. Call ϕ a *restricted Boolean formula* if it meets the
 646 following three additional conditions:

- 647 1. Each variable v_i occurs as a literal in either two clauses or in three clauses. Exactly one of
 648 these occurrences is a negative occurrence (the other one or two are positive occurrences).
- 649 2. Each clause c_j contains either two literals or three literals (these may be any combination
 650 of positive and negative).
- 651 3. The *bipartite variable–clause incidence graph* of ϕ is a planar graph.

652 The bipartite variable–clause incidence graph is an undirected graph (V, E) with vertices $V =$
 653 $\{v_1, \dots, v_n, c_1, \dots, c_m\}$ and edges $E = \{\{v_i, c_j\} : \text{variable } v_i \text{ occurs as a literal in clause } c_j\}$.

654 These additional restrictions streamline the proof of Theorem 6.1 in Appendix C.2, by reducing the
 655 complexity of the reduction mapping Boolean formulas to UPP instances.

656 **Problem xSAT.** Restricted Boolean satisfiability, or xSAT, is a decision problem. The instances are
 657 all *restricted Boolean formulas*. The affirmative instances are all satisfiable formulas.

658 Figure 4 illustrates some instances of Problem **xSAT**.

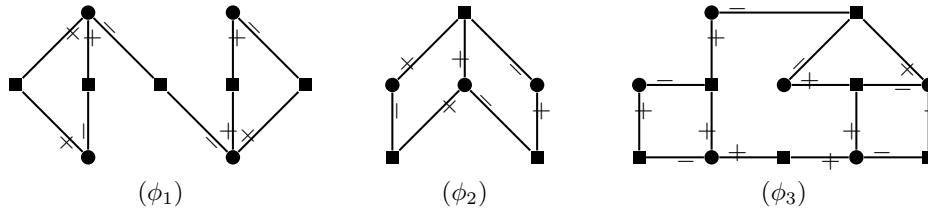


Figure 4: Three example restricted Boolean formulas in conjunctive normal form are as follows:
 $\phi_1 = (v_1 \vee v_2) \wedge (\bar{v}_1 \vee v_2) \wedge (v_3 \vee v_4) \wedge (\bar{v}_3 \vee v_4) \wedge (\bar{v}_2 \vee \bar{v}_4)$; $\phi_2 = (\bar{v}_1 \vee v_2) \wedge (v_1 \vee v_2 \vee \bar{v}_3) \wedge (\bar{v}_2 \vee v_3)$;
 and $\phi_3 = (\bar{v}_1 \vee \bar{v}_3 \vee v_4) \wedge (v_1 \vee \bar{v}_2 \vee v_5) \wedge (v_3 \vee \bar{v}_4 \vee v_6) \wedge (v_2 \vee \bar{v}_5) \wedge (v_5 \vee v_6) \wedge (v_4 \vee \bar{v}_6)$. The
 bipartite variable–clause incidence graphs of the three formulas are depicted above (circles indicate
 variable vertices, squares indicate clause vertices, positive and negative occurrences (edges) are
 marked accordingly. Formula ϕ_1 is unsatisfiable whereas formulas ϕ_2 and ϕ_3 are each satisfiable.

Theorem C.1. *Problem xSAT is \mathcal{NP} -complete.*

Proof. $\text{xSAT} \in \mathcal{NP}$ as since $\text{SAT} \in \mathcal{NP}$. To show $\text{SAT} \rightarrow \text{xSAT}$ much of the work is already done:

1. Cook (1971) reduced SAT to 3-SAT, a variant with at most three literals per clause.
2. Lichtenstein (1982) extended this reduction to planar 3-SAT, a variant with at most three literals per clause and a planar bipartite clause–variable incidence graph (in fact the planarity condition studied by Lichtenstein is even stronger).
3. Cerioli et al. (2004, 2011) extended the reduction to planar 3-SAT₃—a variant of planar 3-SAT with at most three occurrences per variable. Cerioli et al. (2004, 2011) used similar techniques to Tovey (1984), Jansen and Müller (1995), and Berman et al. (2003), who studied variants of SAT with bounded occurrences per variable, but no planarity restriction.

It remains to efficiently construct from an instance ϕ of planar 3-SAT₃ an equisatisfiable formula ϕ' having *additionally* (a) at least two occurrences per variable, (b) at least two variables per clause, and (c) exactly one negative occurrence per variable. This can be achieved by removing variables, literals, and clauses and negating occurrences in ϕ (noting that such operations do not affect the conditions on ϕ) as follows. First, establish (a) and (b)¹¹ by exhaustively applying the following (polynomial-time) operations.

- (i) If a variable always occurs with one sign (including never or once), remove the variable and all incident clauses. The resulting (sub)formula is equisatisfiable: extend a satisfying assignment by satisfying the removed clauses with the removed variable.
- (ii) If a clause contains a single literal, this variable is determined in a satisfying assignment. Remove the variable and clause along with any other clauses in which the variable occurs with that sign. For other occurrences, retain the clause but remove the literal.¹¹ If the resulting formula is unsatisfiable, then the additional variable won't help, and if the resulting formula is satisfiable, then so is the original with the appropriate setting of the variable to satisfy the singleton clause.

Only a polynomial number of operations are possible as each removes one variable. Moreover, thanks to (i), each variable with two occurrences now has one negative occurrence (as required). For variables with three occurrences, one or two are negative. Establish (c) by negating all three occurrences for those that have two negative occurrences (so that the two become positive and the one becomes negative as required). The result is equisatisfiable because satisfying assignments can be translated by negating the truth value assigned to this variable. Carrying out this negation operation for the necessary variables takes polynomial time and completes the reduction. \square

C.2 Complexity of uniform point partition

Theorem 6.1 is a corollary of Theorem B.1 and the following two results.

Theorem C.2. $\text{UPP} \in \mathcal{NP}$.

Proof. An (r, ε) -partition of the points acts as a certificate. Such a partition can be verified in polynomial time by computing the pairwise uniform distances within each group. \square

Theorem C.3. $\text{xSAT} \rightarrow \text{UPP}$.

Proof. This proof is more substantial. We describe the reduction algorithm in detail, and then formally prove its efficiency and correctness, over the remainder of this section (pages 18–24).

Reduction overview. Given an xSAT instance, the idea is to build a UPP instance with a collection of points mirroring the structure of the bipartite variable–clause incidence graph of the restricted Boolean formula. To each variable vertex, clause vertex, and edge corresponds a collection of points. The points for each variable vertex can be partitioned in one of two configurations, based on the value of the variable in a truth assignment. Each determines the available groupings of the incident edge's points so as to propagate these assignments to the clauses. The maximum number of groups is set so that there are enough to include the points of each clause vertex if and only if some variable satisfies that clause in the assignment.

¹¹If a clause contains no literals, whether initially or due to the removal of a literal through operation (ii), then the formula is unsatisfiable. Return any unsatisfiable instance of xSAT , such as ϕ_1 of Figure 4.

707 **Reduction step 1: Lay out the graph on a grid.** Due to the restrictions on the xSAT instance,
708 the bipartite variable–clause incidence graph is planar with maximum degree three. Therefore there
709 exists a graph layout where (1) the vertices are positioned at integer coordinates, and (2) the edges
710 comprise horizontal and vertical segments between adjacent pairs of integer coordinates (Valiant,
711 1981, §IV). Moreover, such a (*planar, rectilinear, integer*) *grid layout* can be constructed in polyno-
712 mial time (see, e.g., Valiant, 1981; Liu et al., 1998; there is no requirement to produce an “optimal”
713 layout—just a polynomial-time computable layout). Figure 5 shows three examples.

714 **Reduction step 2: Divide the layout into tiles.** The grid layout serves as a blueprint for a UPP
715 instance: it governs how the points corresponding to each variable vertex, clause vertex, and edge
716 are arranged in the plane. The idea is to conceptually divide the plane into unit square *tiles*, with
717 one tile for each coordinate of the integer grid occupied by a vertex or edge in the grid layout. The
718 tile divisions for the running examples are shown in Figure 5.

719 Due to the restrictions on xSAT instances, any tile division uses just forty distinct tile *types* (just
720 nine up to rotation and reflection). There are straight edge segments and corner edge segments,
721 plus clause and variable vertices with two or three edges in any direction, and for variable vertices,
722 exactly one direction corresponds to a negative occurrence. Figure 6 enumerates these types.

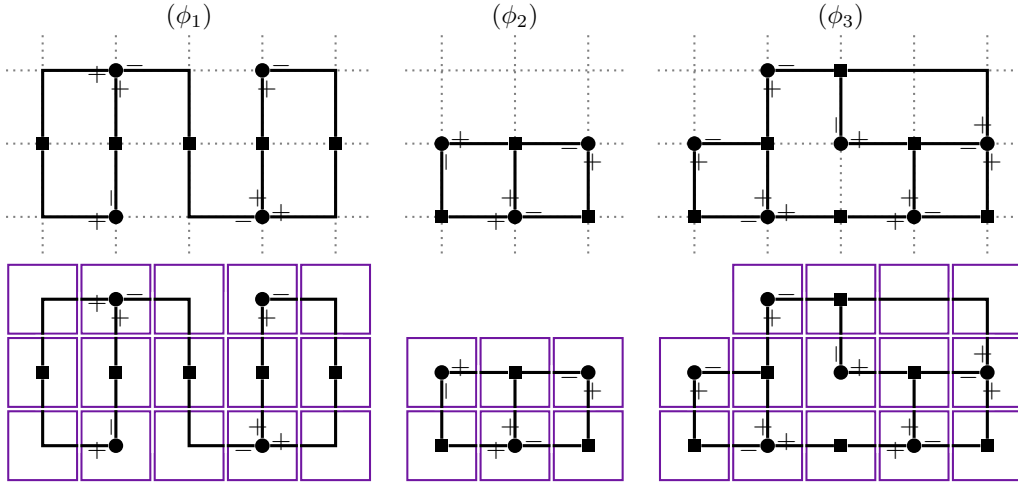


Figure 5: **Top:** Example planar, rectilinear, integer grid layouts of the bipartite variable–clause incidence graphs from Figure 4. Note: these layouts are computed by hand—those produced by standard algorithms may be larger. **Bottom:** Division of the same grid layouts into tiles.

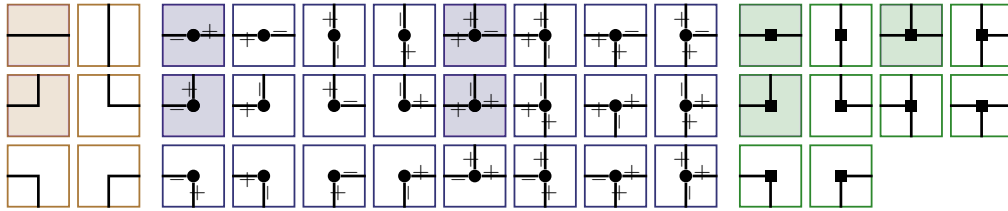


Figure 6: Just forty tile types suffice to construct any tile division of a grid layout. Up to rotation and reflection, just nine distinct types (for example, those highlighted) suffice.

723 **Reduction step 3: Populate the instance with points.** The points of the UPP instance are of two
724 kinds (described in more detail below): (1) *boundary points* between neighbouring pairs of tiles;
725 and (2) *interior points* within each tile in a specific arrangement depending on the tile type. The
726 boundary points can be grouped with interior points of one or the other neighbouring tile. In this
727 way, boundary points couple the choice of how to partition the interior points of neighbouring tiles,
728 creating the global constraint that corresponds to satisfiability.

729 **Reduction step 3a: Boundary points between neighbouring tiles.** There is one boundary point
730 at the midpoint of the boundary between each pair of neighbouring tiles. A pair of *neighbouring*
731 *tiles* is one for which there is an edge crossing the boundary. It is not sufficient for the tiles to be
732 adjacent. Figure 7 clarifies this distinction using the running examples.

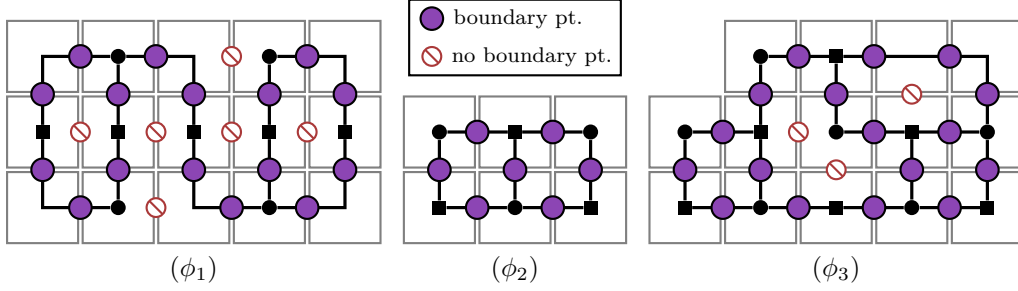


Figure 7: Example of the placement of boundary points between neighbouring tiles. Boundary points are not placed between adjacent tiles if no edge crosses this tile boundary.

733 **Reduction step 3b: Interior points for variable tiles.** Table 1 shows arrangements of interior
734 points for each type of variable tile (up to rotation and reflection). Due to the restrictions on the
735 xSAT instance, each variable tile has one *negative boundary point* and one or two *positive boundary*
736 *point(s)* (corresponding to the variable occurrences). With a given number of groups, the choice of
737 which boundary point(s) to include corresponds to the value of the variable in a truth assignment.
738 *Lemma C.4.* Consider an interior and boundary point arrangement from Table 1 (first 4 rows), or a
739 rectilinear rotation or reflection of such an arrangement. Let $r \in \{2, 3\}$ be the allocated number of
740 groups, and let $\varepsilon \in \mathbb{R}^+$ be the scale.

- 741 (i) There is no (k, ε) -partition of the interior points if $k < r$.
742 (ii) For any (r, ε) -partition of the interior points, the negative boundary point is within uniform
743 distance ε of all points in some group, if and only if (neither of) the positive boundary
744 point(s) are within uniform distance ε of all points in any group.

745 *Proof.* It suffices to consider the arrangements in Table 1 (first 4 rows) because the uniform dis-
746 tance is invariant to rectilinear rotation and reflection. The claims are then verified by exhaustive
747 consideration of all possible partitions of the interior points into at most r groups. \square

748 The partitions indicated in the table are used while constructing a partition of the whole instance
749 given a satisfying assignment. Conversely, no other partitions of the interior points using r groups
750 are possible, except in the fourth row, where other partitions are possible, but, as suffices for the
751 reduction, there are no partitions including both positive and negative boundary points.

752 **Reduction step 3c: Interior points for edge tiles.** Table 1 shows arrangements of interior points
753 for each type of edge tile (up to rotation and reflection). Once the partition of a variable tile includes
754 either the positive boundary point(s) or the negative boundary point, the role of an edge tile is
755 to propagate this choice to the incident clause. These simple point arrangements ensure that the
756 opposite boundary point can be included in a partition of the interior points if and only if the prior
757 boundary point is not (that is, if and only if it was included by the partition of the interior points of
758 the variable tile or, inductively, the previous edge tile).

759 *Lemma C.5.* Consider an interior and boundary point arrangement from Table 1 (last 2 rows), or a
760 rectilinear rotation or reflection of such an arrangement. Let $\varepsilon \in \mathbb{R}^+$ be the scale.

- 761 (i) There is no (k, ε) -partition of the interior points if $k < 2$.
762 (ii) For any $(2, \varepsilon)$ -partition of the interior points, either boundary point is within uniform dis-
763 tance ε of all points in some group, if and only if the other boundary point is not within
764 uniform distance ε of all points in any group.

765 *Proof.* Special case of Lemma C.4. \square

766 The partitions indicated in Table 1 are the only possible $(2, \varepsilon)$ -partitions of the interior points.


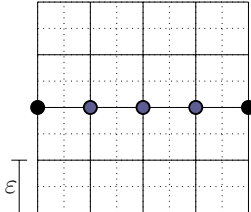
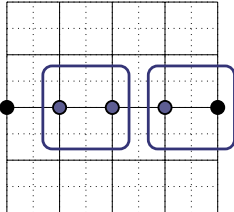
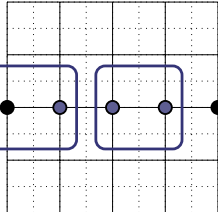

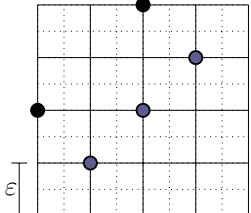
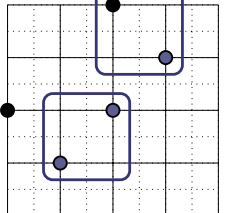
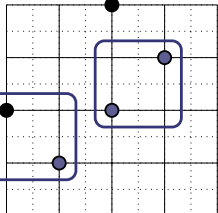

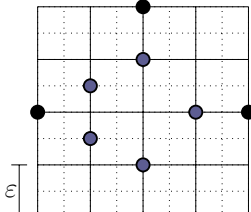
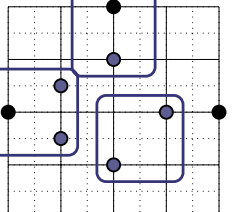
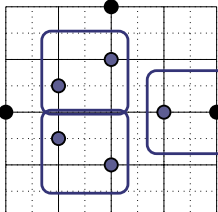

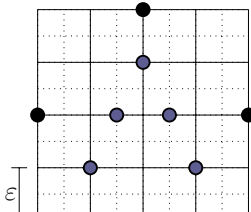
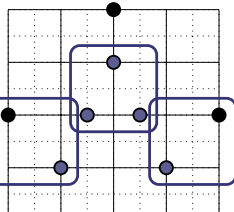
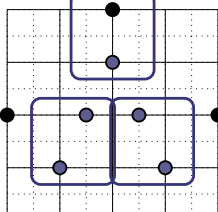

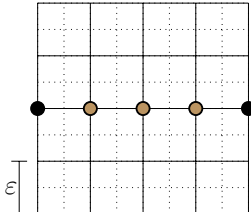
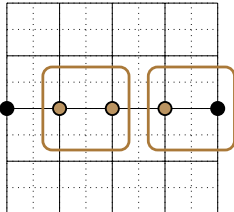
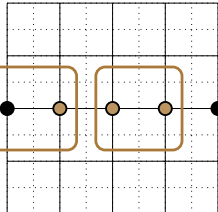

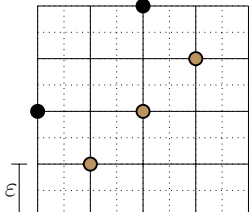
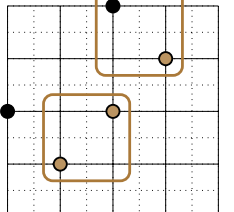
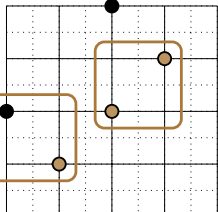
Type	h	r	Arrangement	Possible partitions		
	3	2				
	3	2				
	5	3				
	5	3				
	3	2				
	3	2				

Table 1: Arrangement of interior points for variable tiles (first 4 rows) or edge tiles (last 2 rows). h represents the number of interior points (coloured), with nearby boundary points also shown (black). r represents the number of groups allocated to the tile during the reduction.

767 **Reduction step 3d: Interior points for clause tiles.** Table 2 shows arrangements of interior points
768 for each type of clause tile. The arrangements are such that the interior points of the clause tile can
769 be partitioned if and only if one of the boundary points is not included (that is, it must be included
770 by a neighbouring variable or edge tile, indicating the clause will be satisfied by the corresponding
771 literal).

772 *Lemma C.6.* Consider an interior and boundary point arrangement from Table 2, or a rectilinear
773 rotation or reflection of such an arrangement. Let $r \in \{2, 3\}$ be the allocated number of groups,
774 and let $\varepsilon \in \mathbb{R}^+$ be the scale.

775 (i) There is no (k, ε) -partition of the interior points if $k < r$.

776 (ii) For any (r, ε) -partition of the interior points, there is at least one boundary point that is
777 not within uniform distance ε of all points in any group.

778 *Proof.* Following Lemma C.4, the conditions can be checked exhaustively. \square

779 Table 2 shows the only possible (r, ε) -partitions of the interior points, except in the third row, where
780 a reflected version of the first example partition is also possible.


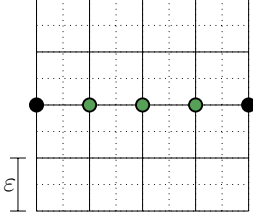
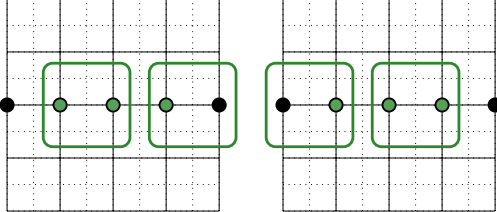

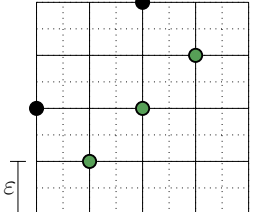
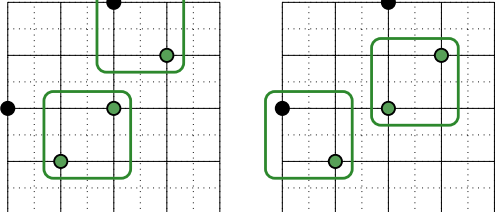

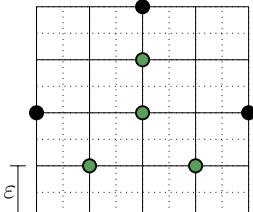
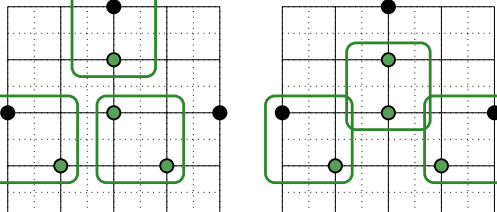
Type	h	r	Arrangement	Possible partitions
	3	2		
	3	2		
	4	3		

Table 2: Arrangement of interior points for clause tiles. The number h represents the number of interior points (coloured). The boundary points are also shown (black). The number r represents the number of groups allocated to the tile during the reduction.

781 **Reduction step 4: Set the number of groups.** For the number of groups, total the allocations for
782 the interior points of each tile (r in Tables 1 and 2). That is, set r for the UPP instance to thrice
783 the number of 3-occurrence variables and 3-literal clauses plus twice the number of edge segments,
784 2-occurrence variables, and 2-literal clauses.

785 **Reduction step 5: Set the uniform diameter.** The reduction works at any (polynomial-time com-
786 putable) scale. For concreteness, set the diameter to $1/4$, giving each tile unit width.

787 **Formal summary of the reduction.** Given an instance of xSAT, that is, a restricted Boolean for-
788 mula ϕ with variables v_1, \dots, v_n and clauses $c_1 \wedge \dots \wedge c_m$, construct an instance of UPP as described
789 in detail in the above steps. Namely, use the points $x_1, \dots, x_h \in \mathbb{R}^2$ as described in Reduction step 3
790 (the interior points from all tiles and the boundary points between neighbouring tiles); a number of
791 groups r as described in Reduction step 4 (the total allocated groups from all of the tiles); and a
792 uniform diameter $\varepsilon = 1/4$ as described in Reduction step 5.

793 Table 3 shows the full UPP instances for the running examples (cf. Figures 4, 5 and 7).

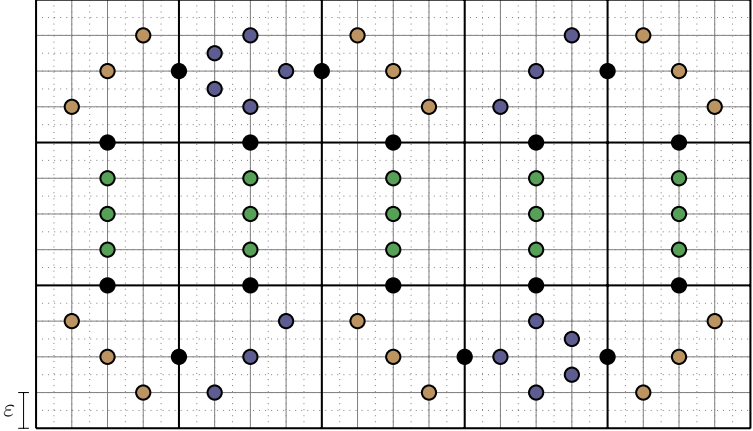
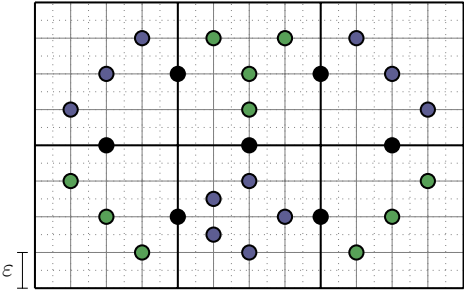
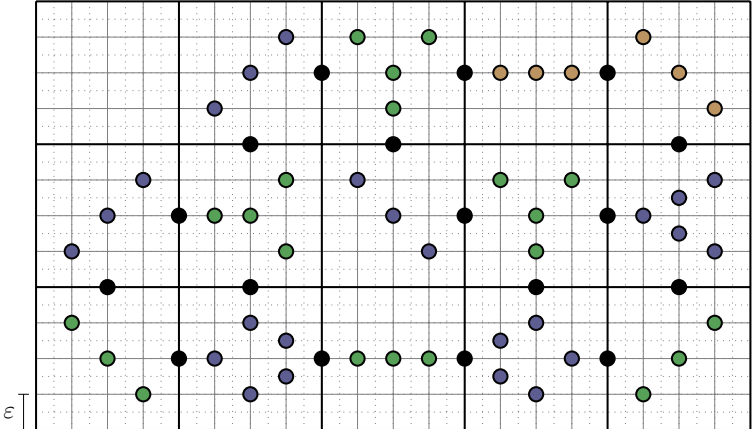
Formula	n	k	Source points
ϕ_1	65	32	
ϕ_2	28	14	
ϕ_3	68	34	

Table 3: Full examples of the reduction from xSAT to UPP, based on xSAT instances described in Figure 4. Exercise: is there an (r, ε) -partition in each case?

Correctness of the reduction. Step 1 (grid layout) runs in polynomial time (Liu et al., 1998), and the remaining steps run in linear or constant time. It remains to show that the constructed instance of UPP is equivalent to the original xSAT instance. That is, we must show that ϕ is satisfiable if and only if there exists an (r, ε) -partition of the points x_1, \dots, x_h .

(\Rightarrow): Suppose ϕ is satisfiable. Let θ be a satisfying truth assignment. Produce an (r, ε) -partition of x_1, \dots, x_h as follows.

1. Partition the interior points of each variable tile as in Table 1. Include the positive boundary point(s) if the variable is assigned “true” in θ , or include the negative boundary point if it is assigned “false”.
2. For each variable tile, follow the *included* boundary point(s) through zero or more edge tiles to the incident clause’s tile, partitioning the interior points of each edge tile according to Table 1 such that the boundary point in the direction of the clause tile is included.
3. Since θ is a satisfying assignment, every clause tile is reached in this way at least once, and thus has at least one of its boundary points included in the groups described so far. For each clause tile, partition the interior points according to Table 2, including the remaining boundary points (if any).
4. For each clause tile, follow the remaining boundary points through zero or more edges back to a variable tile, partitioning the interior points of each edge tile according to Table 1 such that the boundary point in the direction of the variable tile is included.

The final step includes exactly the boundary points of variable tiles that were not included in the first step. Thus, all interior and boundary points are included in some group. The number of groups is exactly in accordance with the allocated number of groups per tile, for a total of r .

(\Leftarrow): Suppose there is an (r, ε) -partition of the points. Observe the following:

- Since the interior points of each tile are separated from the tile boundaries by at least ε , no group can include interior points from two separate tiles.
- It follows that the interior points of each tile must be partitioned into their allocated number of groups. If one tile were to use more groups, some other tile would not get its allocation of groups, and it would be impossible to include all of its interior points in the partition (by Lemmas C.4(i), C.5(i), and C.6(i)).
- Since the boundary points have no allocated groups, each boundary point must be included in a group with interior points from one of its neighbouring tiles.

Now, consider each clause tile. By Lemma C.6(ii), there must be at least one boundary point that is included with the interior points of one of its *neighbouring* tiles. Pick one such direction for each clause and use this to construct a satisfying assignment for ϕ as follows.

In each direction, follow the sequence of zero or more edge tiles back to a variable tile. By Lemma C.5(ii), each boundary point along the sequence of edges must be included with the interior points of the *next* edge tile in the sequence. In turn, the boundary point at the variable tile must be included with the interior points of the variable tile. If this is a positive boundary point, set this variable to “true” in a truth assignment θ , and if it is a negative boundary point, set the variable to “false”.

This uniquely defines the truth assignment θ for all variables reached in this way at least once. If a variable is reached this way from two separate clauses, it must be through its two positive boundary points, since by Lemma C.4(ii), it is impossible for the partition to have both a negative and a positive boundary point included with the interior points of the variable tile. Since some variables may not be reached at all in this way, θ is not completely defined. Complete the definition of θ by assigning arbitrary truth values to such variables.

The truth assignment θ is a satisfying assignment for ϕ . Each clause is satisfied by at least one literal, corresponding to the variable tile that was reached through one of the clause tile’s boundary points not grouped with the clause tile’s interior points in the partition.

This concludes the proof of Theorem C.3. □

D Problem variations and their computational complexity

In this section we discuss several minor variations of Problem [UPC](#).

Uniform vector partition is hard. Consider a generalisation of Problem [UPC](#) beyond the plane, as follows. Let $p \in \mathbb{N}^+$. Given h source vectors $x_1, \dots, x_h \in \mathbb{R}^p$, define an (r, ε) -cover, a list of r covering vectors $y_1, \dots, y_r \in \mathbb{R}^p$ such that the uniform distance between each source vector and its nearest covering vector is at most ε (that is, $\forall i \in \{1, \dots, h\}, \exists j \in \{1, \dots, r\}, \|x_i - y_j\|_\infty \leq \varepsilon$).

Problem UVC^p . Let $p \in \mathbb{N}^+$. Uniform vector cover in \mathbb{R}^p , or UVC^p , is a decision problem. Each instance comprises a collection of $h \in \mathbb{N}$ source points $x_1, \dots, x_h \in \mathbb{R}^p$, a uniform radius $\varepsilon \in \mathbb{R}^+$, and a number of covering points $r \in \mathbb{N}$. The affirmative instances are those for which there exists an (r, ε) -cover of x_1, \dots, x_h .

Theorem D.1. *If $p \geq 2$, then UVC^p is \mathcal{NP} -complete.*

Proof. ($\text{UPC} \rightarrow \text{UVC}^p$): Let $p \geq 2$. Embed the source points from the UPC instance into the first two dimensions of \mathbb{R}^p (leaving the remaining components zero). If there is an (r, ε) -cover of the 2-dimensional points, embed it similarly to derive a p -dimensional (r, ε) -cover. Conversely, if there is a p -dimensional (r, ε) -cover, truncate it to the first two dimensions to derive an (r, ε) -cover.

($\text{UVC}^p \in \mathcal{NP}$): Use an $(r, 2\varepsilon)$ -partition (suitably generalised to p dimensions) as a polynomial-time verifiable certificate. Such a certificate is appropriate along the lines of the proof of Theorem [B.1](#). (An arbitrary (r, ε) -cover is unsuitable as a certificate along the lines of Footnote [8](#).) \square

Remark D.2. By a p -dimensional generalisation of Theorem [B.1](#), p -dimensional generalisations of Problem [UPP](#) and Problem [usgCP](#) are also \mathcal{NP} -complete for $p \geq 2$.

Uniform scalar partition is easy. On the other hand, UVC^1 , which could be called *uniform scalar cover*, is in \mathcal{P} . A minimal cover can be constructed using a greedy algorithm with runtime $\mathcal{O}(h \log h)$. An (r, ε) -cover exists if and only if there are at most r scalars in the result.

Algorithm D.1 (Optimal uniform scalar cover). Proceed:

```

1: procedure UNIFORMSCALARCOVER( $h \in \mathbb{N}, \varepsilon \in \mathbb{R}^+, x_1, \dots, x_h \in \mathbb{R}$ )
2:    $x'_1, \dots, x'_h \leftarrow x_1, \dots, x_h$ , sorted in non-decreasing order.
3:    $j \leftarrow 0$ 
4:   for  $i = 1, \dots, h$  do
5:     if  $j = 0$  or  $x'_i > y_j + \varepsilon$  then
6:        $j \leftarrow j + 1$ 
7:        $y_j \leftarrow x'_i + \varepsilon$ 
8:     end if
9:   end for
10:  return  $y_1, \dots, y_j$ 
11: end procedure
```

Theorem D.3 (Algorithm [D.1](#) correctness). *Let $h \in \mathbb{N}$, $\varepsilon \in \mathbb{R}^+$, and $x_1, \dots, x_h \in \mathbb{R}$. Let $y_1, \dots, y_r = \text{UNIFORMSCALARCOVER}(h, \varepsilon, x_1, \dots, x_h)$. Then (i) y_1, \dots, y_r is an (r, ε) -cover of x_1, \dots, x_h ; and (ii) no (k, ε) -cover exists for $k < r$.*

Proof. (i): After each iteration $i = 1, \dots, h$, if the if branch was entered, then $\|x'_i - y_j\|_\infty = \|x'_i - (x'_i + \varepsilon)\|_\infty = \varepsilon$. If not, that is, if $x'_i \leq y_j + \varepsilon$, let k be the last iteration in which j was increased. Then $x'_i \geq x'_k = y_j - \varepsilon$. In summary, $y_j - \varepsilon \leq x'_i \leq y_j + \varepsilon$, as required.

(ii): For $j = 1, \dots, r$, let $\xi_j = x'_i$ from the iteration in which y_j was defined. Then for $j \neq k$, $\|\xi_j - \xi_k\|_\infty \geq 2\varepsilon$. No covering scalar could be within distance ε of any two of these source scalars. Thus at least r covering scalars are required to cover ξ_1, \dots, ξ_r , and, in turn, x_1, \dots, x_h . \square

Clique partition on ‘square penny’ graphs is hard. [Cerioli et al. \(2004, 2011\)](#) showed that clique partition is \mathcal{NP} -complete in a restricted variant of unit disk graphs called *penny graphs*. A penny graph is a unit disk graph in which the Euclidean distance between source points is *at least* ε , evoking the contact relationships among non-overlapping circular coins.

Our reduction (Appendix [C](#)) happens to produce a set of source points for a unit square graph satisfying a uniform distance version of this condition. Therefore, our proof shows that clique partition remains \mathcal{NP} -complete in this special family of graphs as well.

E A characterisation of the class of bounded-rank parameters

We offer a characterisation of the subset of parameter space with parameters of a given maximum rank. These are the subsets to which detecting proximity is proved \mathcal{NP} -complete in Theorem 6.2.

Let $h, r \in \mathbb{N}$ with $r < h$. The *bounded rank region* of rank r is the subset of parameters of rank at most r , $\mathfrak{B}_r = \{w \in \mathcal{W}_h : \text{rank}(w) \leq r\} \subseteq \mathcal{W}_h$. The key to characterising bounded rank regions is that for each parameter in \mathfrak{B}_r , at least $h - r$ units would be removed during lossless compression. Considering the various possible ways in which units can be removed in the course of Algorithm 4.1 leads to a characterisation of the bounded rank region as a union of linear subspaces.

To this end, let $H = \{1, \dots, h\}$, and define a *compression trace on h units* as a 4-tuple $(\bar{I}, \Pi, \bar{K}, \sigma)$ where $\bar{I} \subseteq H$ is a subset of units (conceptually, those to be removed in Stage 1), $\Pi = \Pi_1, \dots, \Pi_J$ is a partition of $H \setminus \bar{I}$ (the remaining units) into J groups (to be merged in Stage 2), $\bar{K} \subset \{1, \dots, J\}$ (merged units removed in Stage 3), and $\sigma \in \{-1, +1\}^h$ is a sign vector (unit orientations for purposes of merging). The length of the compression trace $(\bar{I}, \Pi, \bar{K}, \sigma)$ on h units is $J - |\bar{K}|$ (representing the number of units remaining). A compression trace of length r thus captures the notion of a “way in which $h - r$ units can be removed in the course of Algorithm 4.1”.

Theorem E.1. *Let $h \in \mathbb{N}$. The bounded rank region $\mathfrak{B}_r \subset \mathcal{W}_h$ is a union of linear subspaces*

$$\mathfrak{B}_r = \bigcup_{(\bar{I}, \Pi, \bar{K}, \sigma) \in \Xi(h, r)} \left(\bigcap_{i \in \bar{I}} S_i^{(1)} \cap \bigcap_{j=1}^J S_{\Pi_j, \sigma}^{(2)} \cap \bigcap_{k \in \bar{K}} S_{\Pi_k, \sigma}^{(3)} \right) \quad (1)$$

where $\Xi(h, r)$ denotes the set of all compression traces on h units with length r ;

$$S_i^{(1)} = \{ (a_1, b_1, c_1, \dots, a_h, b_h, c_h, d) \in \mathcal{W}_h : b_i = 0 \};$$

$$S_{\Pi, \sigma}^{(2)} = \{ (a_1, b_1, c_1, \dots, a_h, b_h, c_h, d) \in \mathcal{W}_h : \forall i, j \in \Pi, \sigma_i b_i = \sigma_j b_j \wedge \sigma_i c_i = \sigma_j c_j \}; \text{ and}$$

$$S_{\Pi, \sigma}^{(3)} = \{ (a_1, b_1, c_1, \dots, a_h, b_h, c_h, d) \in \mathcal{W}_h : \sum_{i \in \Pi} \sigma_i a_i = 0 \}.$$

Proof. (\supseteq): Suppose $w = (a_1, b_1, c_1, \dots, a_h, b_h, c_h, d) \in \mathcal{W}_h$ is in the union in (1), and therefore in the intersection for some compression trace $(\bar{I}, \Pi, \bar{K}, \sigma)$. The constraints imposed on w by membership in this intersection imply that the network is compressible:

1. For $i \in \bar{I}$, since $w \in S_i^{(1)}$, $b_i = 0$, so unit i can be removed.

2. For $j = 1, \dots, J$, since $w \in S_{\Pi_j, \sigma}^{(2)}$, the units in Π_j can be merged together.

3. For $k \in \bar{K}$, since $w \in S_{\Pi_k, \sigma}^{(3)}$, merged unit k has outgoing weight 0 and can be removed.

It follows that there is a parameter with $J - |\bar{K}|$ units that is functionally equivalent to w . Therefore $\text{rank}(w) \leq J - |\bar{K}| = r$ and $w \in \mathfrak{B}_r$.

(\subseteq): Conversely, suppose $w \in \mathfrak{B}_r$. Construct a compression trace following COMPRESS(w). First, set $\sigma_i = \text{sign}(b_i)$ where $b_i \neq 0$ (if $b_i = 0$, set $\sigma_i = \pm 1$ arbitrarily, this has no effect). Then:

1. Set $\bar{I} = \{1, \dots, h\} \setminus I$ where I is computed on line 3. It follows that for $i \in \bar{I}$, $w \in S_i^{(1)}$.

2. Set Π to the partition computed on line 6. It follows that for $j = 1, \dots, J$, $w \in S_{\Pi_j, \sigma}^{(2)}$.

3. Set $\bar{K} = \{j \in \{1, \dots, J\} : \alpha_j = 0\}$ (cf. lines 8,12). Thus for $k \in \bar{K}$, $w \in S_{\Pi_k, \sigma}^{(3)}$.

By construction w is in $\bigcap_{i \in \bar{I}} S_i^{(1)} \cap \bigcap_{j=1}^J S_{\Pi_j, \sigma}^{(2)} \cap \bigcap_{k \in \bar{K}} S_{\Pi_k, \sigma}^{(3)}$. However, the compression trace $(\bar{I}, \Pi, \bar{K}, \sigma)$ has length $\text{rank}(w) \leq r$. If $\text{rank}(w) < r$, remove constraints on $r - \text{rank}(w)$ units by some combination of the following operations: (1) remove one unit from \bar{I} (add it as singleton group to Π), (2) remove one unit from a non-singleton group in Π (add it back to Π as a singleton group), and/or (3) remove one merged unit from \bar{K} . None of these operations add nontrivial constraints on w , so it's still the case that w is in the intersection for the modified compression trace. However, now the length of the compression trace is r , so it follows that w is in the union as required. \square

F Some additional properties of the proximate rank

We document some additional basic properties of the proximate rank.

Variation with uniform radius. Fix $w \in \mathcal{W}_h$. Then $0 \leq \text{prank}_\varepsilon(w) \leq \text{rank}(w)$, depending on ε . The following proposition demonstrates some basic properties of this relationship.

Proposition F.1. Let $h \in \mathbb{N}$. Fix $w \in \mathcal{W}_h$ and consider $\text{prank}_\varepsilon(w)$ as a function of ε . Then,

(i) $\text{prank}_\varepsilon(w)$ is antitone in ε : if $\varepsilon \geq \varepsilon'$ then $\text{prank}_\varepsilon(w) \leq \text{prank}_{\varepsilon'}(w)$; and

(ii) $\text{prank}_\varepsilon(w)$ is right-continuous in ε : $\lim_{\delta \rightarrow 0^+} \text{prank}_{\varepsilon+\delta}(w) = \text{prank}_\varepsilon(w)$.

Proof. For (i), put $u \in \bar{B}_\infty(w; \varepsilon')$ such that $\text{rank}(u) = \text{prank}_{\varepsilon'}(w)$. Then $u \in \bar{B}_\infty(w; \varepsilon)$, so $\text{prank}_{\varepsilon'}(w) = \text{rank}(u) \geq \text{prank}_\varepsilon(w)$. Then for (ii), since $\text{prank}_\varepsilon(w) \leq \text{rank}(w)$ the limit exists by the monotone convergence theorem. Proceed to bound $\text{prank}_\varepsilon(w)$ above and below by $r = \lim_{\delta \rightarrow 0^+} \text{prank}_{\varepsilon+\delta}(w)$. For the lower bound, for $\delta \in \mathbb{R}^+$, $\text{prank}_{\varepsilon+\delta}(w) \leq \text{prank}_\varepsilon(w)$ by (i), so $r = \lim_{\delta \rightarrow 0^+} \text{prank}_{\varepsilon+\delta}(w) \leq \text{prank}_\varepsilon(w)$.

For the upper bound, since the proximate rank is a natural number, the limit is achieved for some positive δ . That is, $\exists \Delta \in \mathbb{R}^+$ such that $\text{prank}_{\varepsilon+\Delta}(w) = r$. Then for $k = 1, 2, \dots$ put $u_k \in \bar{B}_\infty(w; \varepsilon + \Delta/k)$ with $\text{rank}(u_k) = r$. Since $\bar{B}_\infty(w; \varepsilon + \Delta)$ is compact the sequence u_1, u_2, \dots has an accumulation point—call it u . Now, since $u_1, u_2, \dots \in \mathfrak{B}_r$ (parameters of rank at most r , Appendix E), a closed set (by Theorem E.1), the accumulation point $u \in \mathfrak{B}_r$. Thus, $\text{rank}(u) \leq r$. Finally, $u \in \bigcap_{k=1}^\infty \bar{B}_\infty(w; \varepsilon + \Delta/k) = \bar{B}_\infty(w; \varepsilon)$, so $r \geq \text{rank}(u) \geq \text{prank}_\varepsilon(w)$. \square

A similar proof implies that $\text{prank}_\varepsilon(w)$ achieves its upper bound $\text{rank}(w)$ for small enough $\varepsilon > 0$. The lower bound 0 is also achieved; for example for $\varepsilon \geq \|w\|_\infty = \|w - 0\|_\infty$.

Variation with functionally equivalent parameters. The rank of $w \in \mathcal{W}_h$ is defined in terms of f_w , so functionally equivalent parameters have the same rank. This is not necessarily the case for the proximate rank—consider Example F.2. Similar counterexamples hold if a non-uniform metric is used. This is a consequence of the fundamental observation that functionally equivalent parameters may have rather different parametric neighbourhoods.

Example F.2. Let $\varepsilon \in \mathbb{R}^+$. Consider the neural network parameters $w, w' \in \mathcal{W}_2$ with $w = (2\varepsilon, 2\varepsilon, 0, 0, 0, 0, 0)$ and $w' = (2\varepsilon, 2\varepsilon, 0, 5\varepsilon, 0, 0, 0)$. Then for $x \in \mathbb{R}$,

$$\begin{aligned} f_w(x) &= 0 + 2\varepsilon \tanh(2\varepsilon x + 0) + 0 \tanh(0x + 0) \\ &= 0 + 2\varepsilon \tanh(2\varepsilon x + 0) + 0 \tanh(5\varepsilon x + 0) = f_{w'}(x), \end{aligned}$$

but $\text{prank}_\varepsilon(w) = \text{rank}(\varepsilon, \varepsilon, 0; -\varepsilon, \varepsilon, 0; 0) = 0 \neq 1 = \text{prank}_\varepsilon(w')$.

However, functionally equivalent *incompressible* parameters have the same proximate rank.

Proposition F.3. Let $h \in \mathbb{N}$. Consider a permutation $\pi \in S_h$ and a sign vector $\sigma \in \{-1, +1\}^h$. Define $T_{\pi, \sigma} : \mathcal{W}_h \rightarrow \mathcal{W}_h$ such that

$$T_{\pi, \sigma}(a_1, b_1, c_1, \dots, a_h, b_h, c_h, d) = (\sigma_1 a_{\pi(1)}, \sigma_1 b_{\pi(1)}, \sigma_1 c_{\pi(1)}, \dots, \sigma_h a_{\pi(h)}, \sigma_h b_{\pi(h)}, \sigma_h c_{\pi(h)}, d).$$

Then for $w \in \mathcal{W}_h, \varepsilon \in \mathbb{R}^+$, $\text{prank}_\varepsilon(w) = \text{prank}_\varepsilon(T_{\pi, \sigma}(w'))$.

Proof. $T_{\pi, \sigma} : \mathcal{W}_h \rightarrow \mathcal{W}_h$ is an isometry with respect to the uniform distance, so

$$\bar{B}_\infty(T_{\pi, \sigma}(w); \varepsilon) = \{ T_{\pi, \sigma}(u) : u \in \bar{B}_\infty(w; \varepsilon) \}.$$

Moreover, $T_{\pi, \sigma}$ is a symmetry of the parameter–function map (Chen et al., 1993), so it preserves the implemented function and therefore the rank of the parameters in the neighbourhood. \square

Corollary F.4. Let $h \in \mathbb{N}$. Consider two incompressible (minimal) parameters $w, w' \in \mathcal{W}_h$ and a uniform radius $\varepsilon \in \mathbb{R}^+$. If $f_w = f_{w'}$ then $\text{prank}_\varepsilon(w) = \text{prank}_\varepsilon(w')$.

Proof. Since w, w' are functionally equivalent and incompressible (minimal), they are related by a permutation transformation and a negation transformation (Sussmann, 1992). \square

G Hyperbolic tangent networks with multi-dimensional inputs and outputs

In the main paper we consider single-hidden-layer hyperbolic tangent networks with a single input unit and a single output unit. Our algorithms and results generalise to an architecture with multiple input units and multiple output units, with some minor changes.

Consider a family of fully-connected, feed-forward neural network architectures with $n \in \mathbb{N}^+$ input units, $m \in \mathbb{N}^+$ biased linear output units, and a single hidden layer of $h \in \mathbb{N}$ biased hidden units with the hyperbolic tangent nonlinearity. The weights and biases of the network are encoded in a parameter vector in the format $w = (a_1, b_1, c_1, \dots, a_h, b_h, c_h, d) \in \mathcal{W}_h^{n,m} = \mathbb{R}^{(n+m+1)h+m}$, where for each hidden unit $i = 1, \dots, h$ there is an *outgoing weight vector* $a_i \in \mathbb{R}^m$, an *incoming weight vector* $b_i \in \mathbb{R}^n$, and a bias $c_i \in \mathbb{R}$; and $d \in \mathbb{R}^m$ is a vector of output unit biases. Thus each parameter $w \in \mathcal{W}_h^{n,m}$ indexes a function $f_w : \mathbb{R}^n \rightarrow \mathbb{R}^m$ such that $f_w(x) = d + \sum_{i=1}^h a_i \tanh(b_i \cdot x + c_i)$.

The above notation is deliberately chosen to parallel the case $n = m = 1$ considered in the main paper. This makes the generalisation of our results to the case $n, m \geq 1$ straightforward. First, replace all mentions of the scalar incoming and outgoing weights with incoming and outgoing weight *vectors*. It remains to note the following additional changes.

1. The algorithms and proofs commonly refer to the sign of an incoming weight. For $b \in \mathbb{R}^n$ define $\text{sign}(b) \in \{-1, 0, +1\}$ as the sign of the first nonzero component of b , or zero if $b = 0$. Use this generalised sign function throughout when $n > 1$.
2. Sorting the (signed) incoming weight and bias pairs of the hidden units is a key part of Algorithms 4.1 and 4.2. For pairs of the form $(b, c) \in \mathbb{R}^n \times \mathbb{R}$, lexicographically sort by the components of b and then by c .
3. The reducibility conditions were proven by Sussmann (1992) in the case $n \geq 1$ and $m = 1$. The conditions also hold for $m \geq 1$ (see Fukumizu, 1996; or Anonymous, 2023, §A).
4. The construction of the centre of a bounding rectangle in the proof of Theorem 6.2 straightforwardly generalises to the construction of the centre of a bounding right cuboid.

For Theorem 6.2, the proof that $\text{UPC} \rightarrow \text{PR}$ requires no generalisation, because it suffices to construct instances of Problem PR with $n = m = 1$. To see directly that Problem PR remains hard in networks with n input units, compare with UVC^{n+1} (Problem UVC^p discussed in Appendix D; $n + 1$ for n incoming weights plus one bias for each hidden unit).