Path Puzzles: Discrete Tomography with a Path Constraint is Hard

Jeffrey Bosboom* Erik D. Demaine* Martin L. Demaine* Adam Hesterberg*
Roderick Kimball† Justin Kopinsky*

Abstract

We prove that path puzzles with complete row and column information—or equivalently, 2D orthogonal discrete tomography with Hamiltonicity constraint—are strongly NP-complete, ASP-complete, and #P-complete. Along the way, we newly establish ASP-completeness and #P-completeness for 3-Dimensional Matching and Numerical 3-Dimensional Matching.

1 Introduction

Path puzzles are a type of pencil-and-paper logic puzzle introduced in Roderick Kimball's 2013 book [Kim13] and featured in *The New York Times*'s Wordplay blog [Ant14]. Figure 1 gives a small example. A puzzle consists of a (rectangular) grid of cells with two exits (or "doors") on the boundary and numerical constraints on some subset of the rows and columns. A solution consists of a single non-intersecting path which starts and ends at two boundary doors and which passes through a number of cells in each constrained row and column equal to the given numerical clue. Many variations of path puzzles are given in [Kim13] and elsewhere, for example using non-rectangular grids, grid-internal constraints, and additional candidate doors, but these generalizations make the problem only harder.

*Massachusetts Institute of Technology, {jbosboom,edemaine,mdemaine,achester,jkopin}@mit.edu
†Enigami Puzzles & Games

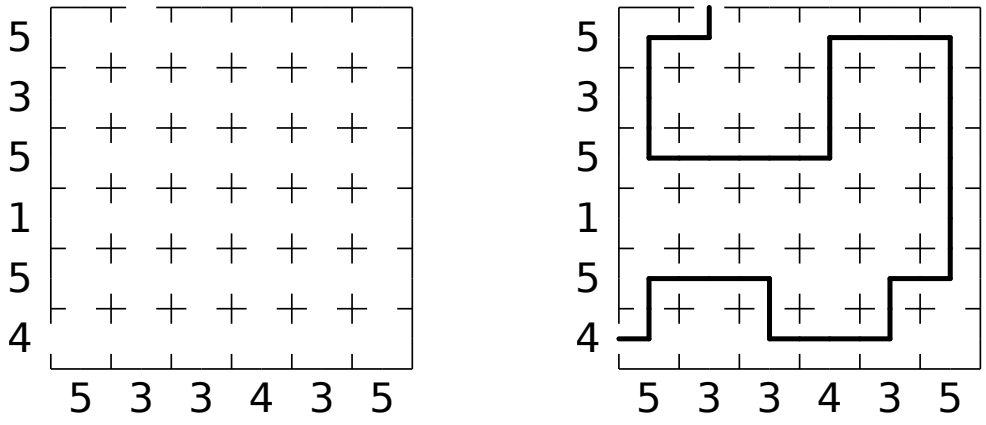

Figure 1: A Path Puzzle with complete row/column information (left) and its solution (right).

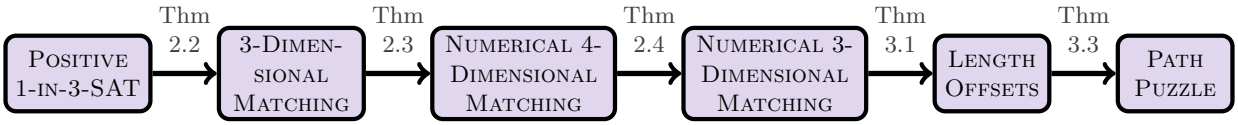

Figure 2: The chain of reductions used in our proof.

Path puzzles are closely related to *discrete tomography* [HK12], in particular the 2D orthogonal form: given the number of black pixels in each row and column, reconstruct a black-and-white image. This problem arises naturally in reconstruction of shapes via x-ray images (which measure density). Vanilla 2-dimensional discrete tomography is known to have efficient (polynomial-time) algorithms [HK12], though it becomes hard under certain connectivity constraints on the output image [DLN12].[1] A path puzzle is essentially a 2-dimensional discrete tomography problem with partial information (not all row and column counts) and an additional Hamiltonicity (single-path) constraint on the output image.

Our results. Unlike 2-dimensional discrete tomography, we show that path puzzles are NP-complete, even with perfect information (i.e., with all row and column counts specified). In other words, 2-dimensional discrete tomography becomes NP-complete with an added Hamiltonicity constraint. In fact, we prove the stronger results that perfect-information path puzzles are Another Solution Problem (ASP) hard and (to count solutions) #P-complete.

Figure 2 shows the chain of reductions we use to prove hardness of Path Puzzle. To preserve hardness for the ASP and #P classes, our reductions are *parsimonious*; that is, they preserve the number of solutions between the source and target problem instances, generally by showing a one-to-one correspondence thereof. We start from the source problem of Positive 1-in-3-SAT which is known to be ASP-hard [Set02,HMRS98] and (to count solutions) #P-complete [HMRS98]. Along the way, we newly establish strong ASP-hardness and #P-completeness for 3-Dimensional Matching, Numerical 4-Dimensional Matching, Numerical 3-Dimensional Matching, and a new problem Length Offsets, in addition to Path Puzzle.

Fonts. To further communicate the challenge of path puzzles to the general public, we designed a mathematical puzzle typeface (as part of a series[2]). Figure 3 gives the puzzle font, which has one path puzzle for each letter of the alphabet. Their solutions are designed to look like the 26 letters of the alphabet, and are verified unique by exhaustive search. Look ahead to the solved font in Figure 8 in Appendix A when you no longer want to solve the puzzles.

2 Numerical 3DM is ASP-Complete and #P-Complete

The goal of this section is to prove that Numerical 3-Dimensional Matching is *strongly* ASP- and #P-complete, i.e., ASP- and #P-complete even when the n numbers are bounded by a polynomial in n. We follow a similar chain of reductions by Garey and Johnson [GJ79], namely 3SAT

[1] Most sets of row and column constraints are ambiguous; constraining the output image makes the problem harder by preventing an easy image from being found instead.

[2] See http://erikdemaine.org/fonts

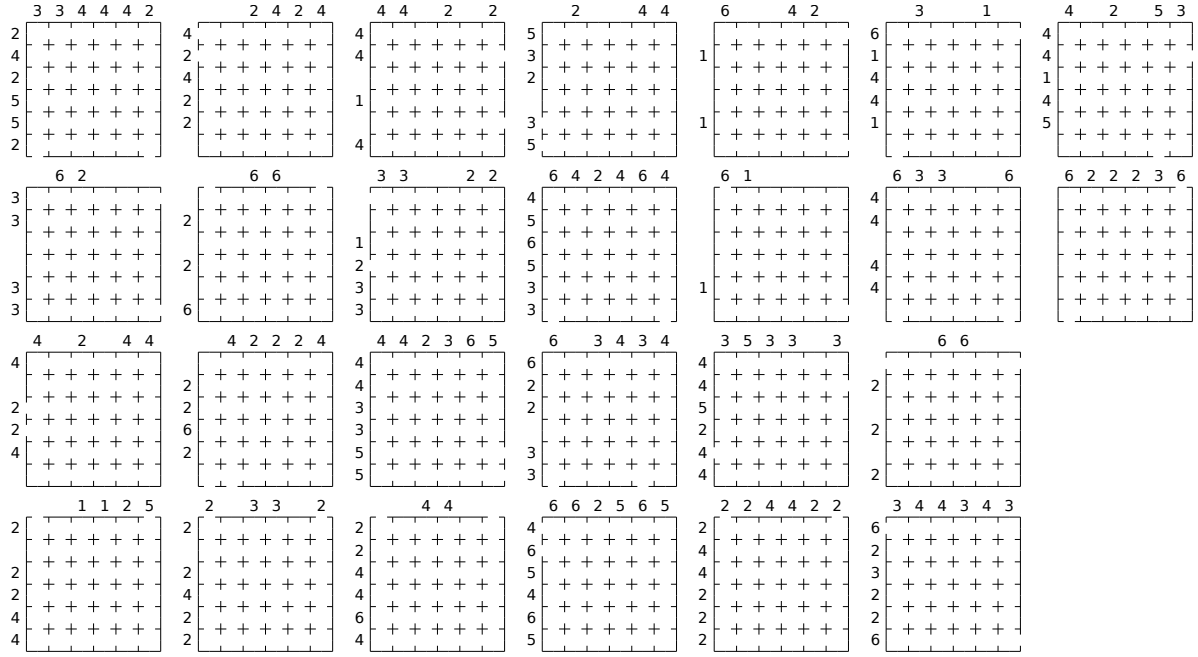

Figure 3: Puzzle font

$\rightarrow$ 3-Dimensional Matching $\rightarrow$ 4-Partition $\rightarrow$ 3-Partition, but replacing k-Partition with Numerical k-Dimensional Matching and starting from a different version of SAT:

Problem 2.1 (Positive 1-in-3-SAT)**.** *Given a 3CNF formula C with only positive literals, is there an assignment of variables such that exactly one literal in each clause of C is true?*

Lemma 2.1. Positive 1-in-3-SAT *is ASP-hard and #P-hard.*

Proof. 3SAT is shown to be #P-hard in [Val79]. Section 3.2.1 of [Set02] shows that 3SAT is ASP-hard.[3] Theorem 3.8 of [HMRS98] gives a parsimonious reduction from 3SAT to Positive 1-in-3-SAT.[4] Combining these results gives the claim. □

Problem 2.2 (3-Dimensional Matching)**.** *Given three sets X, Y, Z of equal cardinality and a set T of triples (x, y, z) where $x \in X, y \in Y, z \in Z$, is there a set $S \subseteq T$ such that each element of X, Y, Z appears in exactly one triple in S?*

Theorem 2.2. 3-Dimensional Matching *is ASP-hard and #P-hard, even when T is constrained not to contain any two triples agreeing on more than one coordinate.*

Proof. We give a parsimonious reduction from Positive 1-in-3-SAT, using the variable gadget from Garey and Johnson's reduction from 3-SAT to 3-Dimensional Matching [GJ79, Thm. 3.2, p. 50]. Given a Positive 1-in-3-SAT instance with a set V of variables and C of clauses, we construct the corresponding 3-Dimensional Matching instance as follows. We will represent

[3]Section 3.2.4 of [Set02] proves that 1-in-3-SAT is ASP-hard. Unfortunately, their problem definition allows negative clauses, while we need Positive 1-in-3-SAT.

[4]In [HMRS98], Positive 1-in-3-SAT is called "1-Ex3MonoSat".

the 3-DIMENSIONAL MATCHING instance as a hypergraph that is tripartite and 3-uniform, i.e., in which each edge connects exactly three vertices of different colors according to a 3-coloring of the vertices. We will say that vertices colored 0 belong to X, vertices colored 1 belong to Y, and vertices colored 2 belong to Z.

Clause triplication. First we triplicate each clause, producing the multiset $C' = C \sqcup C \sqcup C$ (the disjoint union of three copies of C). As a result, the number n_x of occurrences of each variable $x \in V$ in clauses in C' (multiply counting if x occurs multiple times in the same clause) is divisible by 3. A truth assignment for V satisfies C' if and only if it satisfies C, so this triplication does not affect correctness, but it will help us obtain a 3-coloring.

Variable gadget. Next, for each variable $x \in V$, we create a *variable gadget* consisting of $4n_x$ vertices associated with x; refer to Figure 4. We call n_x of the vertices *positive* x vertices, denoted $x_0, x_1, \ldots, x_{n_x-1}$ (one for each occurrence of x in C'); we call n_x of the vertices *negative* x vertices, denoted $\bar{x}_0, \bar{x}_1, \ldots, \bar{x}_{n_x-1}$; and we call $2n_x$ of them *auxiliary* vertices, denoted $x'_0, x'_1, \ldots, x'_{2n_x-1}$. The edges covering the auxiliary vertices are as follows: for each $i \in \{0, 1, \ldots, n_x - 1\}$, we add the "positive" edge $(x_i, x'_{2i}, x'_{2i+1})$ and the "negative" edge $(\bar{x}_i, x'_{2i+1}, x'_{(2i+2) \bmod 2n_x})$. No other edges cover the auxiliary vertices, so there are only two ways to cover them: choose all the positive edges, thereby covering all the positive vertices and none of the negative vertices, or choose all the negative edges, thereby covering all the negative vertices and none of the positive vertices. The former choice represents assigning x to be TRUE, while the latter choice represents assigning x to be FALSE.

Because n_x is divisible by 3, we can 3-color the vertices by assigning colors $0, 1, 2, 0, 1, 2, \ldots$ to the auxiliary vertices $x'_0, x'_1, x'_2, \ldots$, then coloring each positive and negative vertex with the one color not used in its edge, as shown in Figure 4.

The final edges of the variable gadget serve to collect "garbage" negative vertices. For each $i \in \{0, 1, \ldots, n_x/3-1\}$ (using that n_x is divisible by 3), we add another positive edge $(\bar{x}_{3i}, \bar{x}_{3i+1}, \bar{x}_{3i+2})$. These positive edges overlap the negative edges, so cannot be chosen in the FALSE assignment, but do not overlap the positive edges, and then they cover all the negative vertices. No other edges cover the negative vertices, so again we must choose all the positive edges or all the negative edges.

Therefore, local to the variable gadget, we cover all the auxiliary and negative x vertices, and either all or none of the positive x vertices. Only the positive x vertices will interact with other gadgets, through clause gadgets.

Clause gadget. Finally, for each clause $c = \langle x, y, z \rangle \in C'$, we *identify* one positive x vertex, one positive y vertex, and one positive z vertex all of the same color, resulting in a single vertex covered by one positive edge from each of the three corresponding vertex gadgets; refer to Figure 5. The three identified vertices are chosen to be unique to this clause gadget, so they will not be identified again, and thus will be covered exactly once if and only if exactly one of the three variables is assigned TRUE.

For each of the three copies of a clause in C, we choose the identified vertices to be a different color among $\{0, 1, 2\}$, so that each clause in C consumes exactly one positive vertex of each color from each of the three variable gadgets. (When a variable appears twice in the same clause, two of these variable gadgets will actually be the same, and we will end up consuming two positive

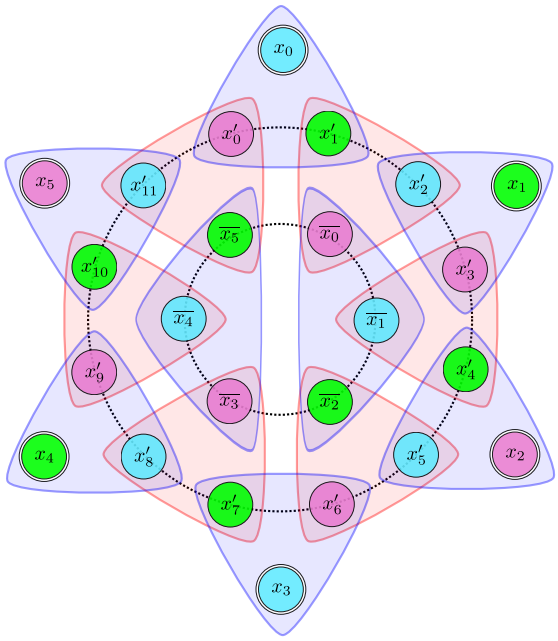

Figure 4: 3DM variable gadget for variable x occurring in $n_x = 6$ clauses in C' (two clauses in C). Hyperedges are drawn as shaded triangles; any solution must include all the positive (blue) or all the negative (red) hyperedges. Vertex colors $0, 1, 2$ are drawn as magenta, green, and cyan. Only the positive vertices (drawn with doubled outlines) are attached to other gadgets.

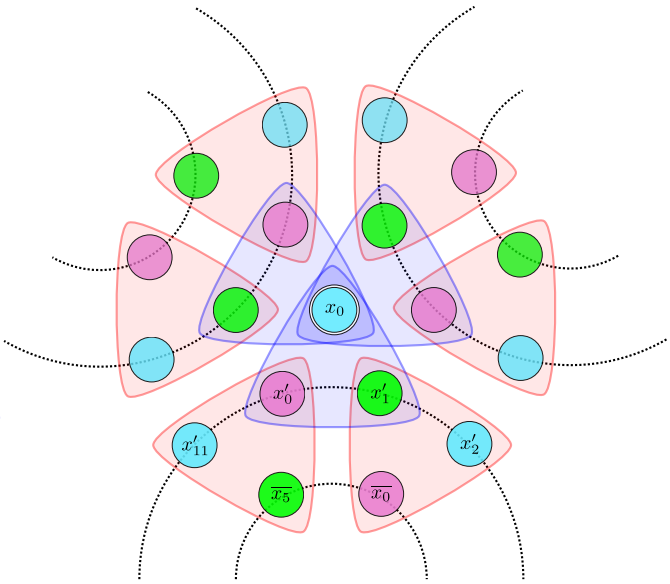

Figure 5: 3DM clause gadget, identifying cyan positive vertices from three variable gadgets (Figure 4). Although here we draw the three variables as distinct, we may also identify positive vertices from the same variable gadget (when the same variable appears twice in the same clause).

vertices of each color, but the accounting remains the same.) Thus we will be able to use each positive vertex in each variable gadget exactly once, without running out of any particular color.

Equivalence. Because the identified (x, y, z) vertex in a clause gadget must be covered by exactly one edge in the 3-Dimensional Matching problem, exactly one of x, y, z must have an assignment of true, which is the Positive 1-in-3-SAT constraint. Thus, given a solved instance of 3-Dimensional Matching, we can extract exactly one solution to the original Positive 1-in-3-SAT instance. Furthermore, given a solution to the Positive 1-in-3-SAT instance, we can produce exactly one solution to the 3-Dimensional Matching instance by choosing all the positive x edges if x is set to true and all the negative edges if x is set to false. Thus the reduction is parsimonious.

Examining Figures 4 and 5, we also see that no hyperedge shares more than one vertex, as claimed. □

Problem 2.3 (Numerical k-Dimensional Matching)**.** *Given k multisets of positive integers $X_1, \ldots, X_k$ and a positive integer target sum t, does there exist a set $S \subseteq X_1 \times \cdots \times X_k$ of k-tuples such that, for each $(x_1, \ldots, x_k) \in S$, $x_1 + \cdots + x_k = t$, and each element of each X_i appears as the ith coordinate in exactly one element of S? (Thus $|X_1| = \cdots = |X_k| = |S|$, and we denote this*

common size by n.)

We will consider specially the cases $k = 3$ and $k = 4$ for which we label the sets X, Y, Z and W, X, Y, Z respectively.

Theorem 2.3. NUMERICAL 4-DIMENSIONAL MATCHING *is strongly ASP-hard and #P-hard, even if* $Y \cup (Y + Z)$ *(where* $Y + Z = \{y + z : y \in Y, z \in Z\}$*) is guaranteed to be a set (not a multiset).*

Proof. We give a parsimonious reduction from 3-DIMENSIONAL MATCHING where no two triples in T agree on more than one coordinate, as guaranteed by Theorem 2.2. Our reduction loosely follows Garey and Johnson's original reduction [GJ79, Thm. 4.3, p. 97] with extra care to ensure parsimony.

We are given a 3-DIMENSIONAL MATCHING instance with elements partitioned into sets

$$X = \{x_1, \ldots, x_n\}, Y = \{y_1, \ldots, y_n\}, Z = \{z_1, \ldots, z_n\}$$

and a set of triples $T \subseteq X \times Y \times Z$. Let $m_T(x_i)$ be the multiplicity of x_i in T, that is, the number of triples of T where x_i is the first coordinate, and similarly define $m_T(y_j)$ and $m_T(z_k)$.

First we pick a large base $B = 100n$. We use the notation $(d_5, d_4, d_3, d_2, d_1, d_0)_B$ to represent the base-B number equal to $\sum_{i=0}^{5} d_i B^i = d_5B^5 + d_4B^4 + d_3B^3 + d_2B^2 + d_1B + d_0$. In the discussion that follows, we use that B is large enough that addition with a digit of the base-B representation of the numbers in question will never carry over to another digit, so $(d_5, d_4, d_3, d_2, d_1, d_0)_B + (d'_5, d'_4, d'_3, d'_2, d'_1, d'_0)_B = (d_5 + d'_5, d_4 + d'_4, d_3 + d'_3, d_2 + d'_2, d_1 + d'_1, d_0 + d'_0)_B$.

We construct a NUMERICAL 4-DIMENSIONAL MATCHING instance with target $t = (40, 0, 0, 0, 0, 0)_B$ and multisets W', X', Y', Z' having the following elements (also refer to Table 1):

(a) For each $x_i \in X$, place $(10, i, 0, 0, 0, 0)_B$ in W', $(11, 0, -i, 0, 0, 0)_B$ in Z', and $m_T(x_i) - 1$ copies of $(10, i, -i, 0, 0, 0)_B$ in W'.

(b) For each $y_j \in Y$, place $(10, 0, 0, j, 0, 0)_B$ in X', $(12, 0, 0, 0, -j, 0)_B$ in W', and $m_T(y_j) - 1$ copies of $(10, 0, 0, j, -j, 0)_B$ in X'.

(c) For each $z_k \in Z$, place $(10, 0, 0, 0, 0, k)_B$ in Y' and $(7, 0, 0, 0, 0, -k)_B$ in X'.

(d) For each $(x_i, y_j, z_k) \in T$, place $(10, -i, 0, -j, 0, -k)_B$ in Z' and $(10, 0, i, 0, j, k)_B$ in Y'.

Importantly, the x_i, y_j, z_k above are indexed starting at 1, not 0. One can verify that the only quadruples summing to t are the following (given in W', X', Y', Z' order):

Type 1. For $(x_i, y_j, z_k) \in T$:

$$\begin{array}{r} (10, \quad i,0, \quad 0,0, \quad 0)_B \\ +(10, \quad 0,0, \quad j,0, \quad 0)_B \\ +(10, \quad 0,0, \quad 0,0, \quad k)_B \\ +(10, -i,0, -j,0, -k)_B \\ =(40, \quad 0,0, \quad 0,0, \quad 0)_B \end{array}$$

Type 2. For $(x_i, y_j, z_k) \in T$:

$$\begin{array}{r} (12,0, \quad 0,0, -j, \quad 0)_B \\ +(\ 7,0, \quad 0,0, \quad 0, -k)_B \\ +(10,0, \quad i,0, \quad j, \quad k)_B \\ +(11,0, -i,0, \quad 0, \quad 0)_B \\ =(40,0, \quad 0,0, \quad 0, \quad 0)_B \end{array}$$

Type 3. For $(x_i, y_j, z_k) \in T$:

$$\begin{array}{r} (10, \quad i, -i, \quad 0, \quad 0, \quad 0)_B \\ +(10, \quad 0, \quad 0, \quad j, -j, \quad 0)_B \\ +(10, \quad 0, \quad i, \quad 0, \quad j, \quad k)_B \\ +(10, -i, \quad 0, -j, \quad 0, -k)_B \\ =(40, \quad 0, \quad 0, \quad 0, \quad 0, \quad 0)_B \end{array}$$

Furthermore, there is a one-to-one correspondence between solutions to the source instance and solutions to the constructed instance by choosing a triple to be in the solution $S \subseteq T$ of the 3-DIMENSIONAL MATCHING instance if and only if both the corresponding triples of the types 1 and 2

	$x_i \in X$	$y_j \in Y$	$z_k \in Z$	$(x_i, y_j, z_k) \in T$
W'	$(10, i, 0, 0, 0, 0)_B$ $\boldsymbol{(10, i, -i, 0, 0, 0)_B}$	$(12, 0, 0, 0, -j, 0)_B$		
X'		$(10, 0, 0, j, 0, 0)_B$ $\boldsymbol{(10, 0, 0, j, -j, 0)_B}$	$(7, 0, 0, 0, 0, -k)_B$	
Y'			$(10, 0, 0, 0, 0, k)_B$	$(10, 0, i, 0, j, k)_B$
Z'	$(11, 0, -i, 0, 0, 0)_B$			$(10, -i, 0, -j, 0, -k)_B$

Table 1: The constructions of W', X', Y', Z'. Each column represents the source of the constructed elements from the original 3-Dimensional Matching instance. Most elements have multiplicity 1; bold elements have multiplicity one fewer than the corresponding 3-Dimensional Matching source item.

above are included in the solution S' of the constructed Numerical 4-Dimensional Matching instance. If a type-1 quadruple is included in the solution for some $(x_i, y_j, z_k) \in T$, then the corresponding type-2 quadruple must also be included because there is no other way to cover the element $(10, 0, i, 0, j, k)_B \in Y'$, and similarly for the reverse. To confirm that the rest of the elements can be covered by the type-3 quadruples, notice that there are exactly the correct number of $(10, i, -i, 0, 0, 0)_B \in W'$ and $(10, 0, 0, j, -j, 0) \in X'$ elements. In particular, for every i, there are $m_T(x_i)$ triples of the form $(10, -i, 0, *, 0, *)_B \in Z'$ and exactly one of these is covered by a type-1 quadruple, so the remaining ones can be matched with the $m_T(x_i) - 1$ elements of the form $(10, i, -i, 0, 0, 0)_B \in W'$, and similarly for the y_j. Thus we have a parsimonious reduction from 3-Dimensional Matching to Numerical 4-Dimensional Matching.

We can verify the claim that $Y' \cup (Y' + Z')$ is a set (not a multiset) using the initial assumption that no two triples in T agree on more than one coordinate. First, Y' is a set because, for each z_k, there is exactly one element $(10, 0, 0, 0, 0, k)_B \in Y'$, and for each triple $(x_i, y_j, z_k) \in T$, there is exactly one element $(10, 0, i, 0, j, k)_B \in Y'$; these two types of elements are disjoint because the third and fifth digits are always zero in the former but nonzero in the latter. Similarly, Z' is a set (a fact we will need later). Also, Y' and $Y' + Z'$ are disjoint because the first digit of any element of Y' is 10 while the first digit of any element of $Y' + Z'$ is at least 20.

To see that $Y' + Z'$ is a set, consider two equal sums $s_1 = y'_1 + z'_1$ and $s_2 = y'_2 + z'_2$ for $y'_1, y'_2 \in Y'$ and $z'_1, z'_2 \in Z'$. From $s_1 = s_2$, it follows that $y'_2 - y'_1 = z'_2 - z'_1$. We claim that, if $s_1 = s_2$, then $z'_1 = z'_2$ and thus $y'_1 = y'_2$, which suffices because we argued that Y' and Z' are sets. To prove the claim, we have two cases, one for each type of element of Z':

Case 1: If $z'_1 = (11, 0, -i_1, 0, 0, 0)_B$, then $z'_2 = (11, 0, -i_2, 0, 0, 0)$ or else s_1 and s_2 would differ in the first digit. Thus $y'_2 - y'_1 = z'_2 - z'_1 = (0, 0, i_1 - i_2, 0, 0, 0)_B$, but by the assumption that there are no two distinct triples of T sharing both y_j and z_k, there are no two distinct elements of Y' whose last two digits are equal but whose third digits are not, so this difference is impossible unless $y'_1 = y'_2$ and $z'_1 = z'_2$.

Case 2: If $z'_1 = (10, -i_1, 0, -j_1, 0, -k_1)_B$, then $z'_2 = (10, -i_2, 0, -j_2, 0, -k_2)_B$ or else s_1 and s_2 would differ in the first digit. Thus $y'_2 - y'_1 = z'_2 - z'_1 = (0, i_1 - i_2, 0, j_1 - j_2, 0, k_1 - k_2)_B$, but no elements of Y' have nonzero second or fourth digits, so it must be that $i_1 = i_2$ and $j_1 = j_2$. By the assumption that no distinct triples of T share both x_i and y_j, it must be that $k_1 = k_2$ as well, so

$z'_1 = z'_2$ as claimed. □

Theorem 2.4. Numerical 3-Dimensional Matching *is strongly ASP-hard and #P-hard, even if X is required to be a set (not a multiset).*

Proof. We give a parsimonious reduction from Numerical 4-Dimensional Matching where $W \cup (W + X)$ is a set, as guaranteed by Theorem 2.3 (relabelling Y, Z to W, X). Our reduction is essentially Garey and Johnson's reduction from 4-Partition to 3-Partition [GJ79, Thm. 4.4, p. 99], with some extra care regarding identical elements and splitting elements into separate sets X', Y', Z'.

Following the reduction in [GJ79], given a Numerical 4-Dimensional Matching instance $W = \{w_1, \ldots, w_n\}$, $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_n\}$, $Z = \{z_1, \ldots, z_n\}$ with target t, we will construct a Numerical 3-Dimensional Matching instance X', Y', Z' with target sum $t' = (64, 4)_B$ in base $B = t$. We assume without loss of generality that every element of W, X, Y, Z is strictly between $t/5$ and $t/3$.[5] First we define the elements that will appear in $X' \cup Y' \cup Z'$:

$$\begin{aligned} w'_i &= (21, 4w_i + 1)_B, \\ x'_j &= (19, 4x_j + 1)_B, \\ y'_k &= (19, 4y_k + 1)_B, \\ z'_\ell &= (21, 4z_\ell + 1)_B, \\ u[w_i, x_j] &= (24, -4(w_i + x_j) + 2)_B, \\ \bar{u}[w_i, x_j] &= (20, \quad 4(w_i + x_j) + 2)_B, \\ C &= (20, 0)_B. \end{aligned}$$

Now we can construct the desired Numerical 3-Dimensional Matching instance, splitting these elements into three multisets X', Y', Z':

$$\begin{aligned} X' &= \{w'_i : 1 \le i \le n\} \cup \{\bar{u}[w_i, x_j] : 1 \le i, j \le n\}, \\ Y' &= \{x'_j : 1 \le j \le n\} \cup \{z'_\ell : 1 \le \ell \le n\} \cup \{n^2 - n \text{ copies of } C\}, \\ Z' &= \{u[w_i, x_j] : 1 \le i, j \le n\} \cup \{y'_k : 1 \le k \le n\}. \end{aligned}$$

There are $2 \times 3 \times 2 = 12$ possible forms of triples, shown below grouped by the equivalence classes modulo 4 of the second coordinate of their sum (with shaded boxes to indicate the only triples that will turn out to be valid):

0 (mod 4)	**1 (mod 4)**	**2 (mod 4)**	**3 (mod 4)**
$(w'_{i'},\ x'_{j'},\ u[w_i, x_j])$	$(\bar{u}[w_{\bar{\imath}}, x_{\bar{\jmath}}],\ x'_{j'},\ u[w_i, x_j])$	$(w'_{i'},\ C,\ y'_{k'})$	$(w'_{i'},\ x'_{j'},\ y'_{k'})$
$(w'_{i'},\ z'_{\ell'},\ u[w_i, x_j])$	$(\bar{u}[w_{\bar{\imath}}, x_{\bar{\jmath}}],\ z'_{\ell'},\ u[w_i, x_j])$		$(w'_{i'},\ z'_{\ell'},\ y'_{k'})$
$(\bar{u}[w_{\bar{\imath}}, x_{\bar{\jmath}}],\ x'_{j'},\ y'_{k'})$			$(w'_{i'},\ C,\ u[w_i, x_j])$
$(\bar{u}[w_{\bar{\imath}}, x_{\bar{\jmath}}],\ z'_{\ell'},\ y'_{k'})$			$(\bar{u}[w_{\bar{\imath}}, x_{\bar{\jmath}}],\ C,\ y'_{k'})$
$(\bar{u}[w_{\bar{\imath}}, x_{\bar{\jmath}}],\ C,\ u[w_i, x_j])$			

[5] If a Numerical 4-Dimensional Matching instance has any elements $\ge t$, it trivially has no solutions (as all elements are positive). Otherwise, we can convert it to an instance with this property by adding $2t$ to each element in W, X, Y, Z and changing the target sum from t to $\hat{t} = 9t$. Then every element is strictly between $2t$ and $3t$, and thus strictly between $\hat{t}/5 = 9t/5$ and $\hat{t}/3 = 9t/3$.

The second coordinate of t' is congruent to 0 (mod 4), so triples in the second, third, and fourth columns never sum to t'.

Of the triples in the first column, two of them cannot actually sum to t'. Triples of the form $(w'_{i'}, z'_{\ell'}, u[w_i, x_j])$ sum to $(66, 4(w_{i'} + z_{\ell'} - w_i - x_j) + 4)_B$. For this to equal t', the second coordinate must equal $-2t+4$, so $w_{i'} + z_{\ell'} - w_i - x_j$ must equal $-t/2$. But by the assumption that every element of W, X, Y, Z is strictly between $t/5$ and $t/3$, the smallest possible value for $w_{i'} + z_{\ell'} - w_i - x_j$ is greater than $-4t/15$, so triples of this form never sum to t'. Similarly, triples of the form $(\bar{u}[w_{\bar{\imath}}, x_{\bar{\jmath}}], x'_{j'}, y'_{k'})$ sum to $(58, 4(w_{\bar{\imath}} + x_{\bar{\jmath}} + x_{j'} + y_{k'}) + 4)_B$. For this to equal t', the second coordinate must equal $6t + 4$, so $w_{\bar{\imath}} + x_{\bar{\jmath}} + x_{j'} + y_{k'}$ must sum to $3t/2$, but the largest possible value for that expression is less than $4t/3$, so triples of this form never sum to t'.

This leaves three forms of triples that can sum to $t' = (64, 4)_B$:

$$(w'_{i'}, x'_{j'}, u[w_i, x_j]), \quad (\bar{u}[w_{\bar{\imath}}, x_{\bar{\jmath}}], z'_{\ell'}, y'_{k'}), \quad \text{and} \quad (\bar{u}[w_{\bar{\imath}}, x_{\bar{\jmath}}], C, u[w_i, x_j]).$$

A triple of the second form encodes a quadruple in the input Numerical 4-Dimensional Matching instance; the triple sums to t' (after a carry $(60, 4t + 4)_B = (64, 4)_B$) exactly when $w_{\bar{\imath}} + x_{\bar{\jmath}} + z_{\ell'} + y_{k'} = t$. The map $w_{\bar{\imath}} + x_{\bar{\jmath}} \mapsto \bar{u}[w_{\bar{\imath}}, x_{\bar{\jmath}}]$ is one-to-one, so from our assumption that $W + X$ is a set, $\{\bar{u}[w_i, x_j]\}$ is also a set, and so this encoding is unique. A triple of the first form sums to t' exactly when $w_{i'} + x_{j'} - w_i - x_j = 0$. Because $W + X$ is a set and the map $w_i + x_j \mapsto u[w_i, x_j]$ is one-to-one, we must have $i' = i$ and $j' = j$ in valid triples of the first form, uniquely collecting the $u[w_i, x_j]$ elements corresponding to $\bar{u}[w_{\bar{\imath}}, x_{\bar{\jmath}}]$ elements used in triples of the second form. Similarly, $\bar{\imath} = i$ and $\bar{\jmath} = j$ in valid triples of the third form, uniquely collecting the unused $u[w_i, x_j]$ and $\bar{u}[w_i, x_j]$ elements using all $n^2 - n$ copies of C. Thus there is a one-to-one correspondence between solutions to the input Numerical 4-Dimensional Matching instance and the constructed Numerical 3-Dimensional Matching instance, so the reduction is parsimonious and Numerical 3-Dimensional Matching is ASP- and #P-hard.

It remains to verify that $X' = \{w'_i\} \cup \{\bar{u}[w_i, x_j]\}$ is a set (not a multiset). We argued above that $\{\bar{u}[w_i, x_j]\}$ is a set (using that $W + X$ is a set), and $\{w'_i\}$ is a set because we assumed W is a set and the map $w_i \mapsto w'_i = (20, 4w_i + 1)_B$ is one-to-one. It remains to show that $\{w'_i\}$ is disjoint from $\{\bar{u}[w_i, x_j]\}$, which follows because $w'_i \equiv 1 \pmod 4$ and $\bar{u}[w_i, x_j] \equiv 2 \pmod 4$. Therefore X' is a set. □

3 Parsimonious Reductions from Numerical 3DM to Path Puzzles

The goal of this section is to parsimoniously reduce Numerical 3-Dimensional Matching (as analyzed in Section 2) to Path Puzzle, thereby proving the latter strongly NP-, ASP-, and #P-hard. We first introduce a more geometric view of Numerical 3-Dimensional Matching, called Length Offsets, and prove its equivalence. It will then be relatively easy to represent Length Offsets as a Path Puzzle.

Problem 3.1 (Length Offsets)**.** *Given a set (not a multiset) of positive integer lengths $a_1, a_2, \ldots, a_n$, and given m nonnegative integer target densities $t_0, t_1, \ldots, t_{m-1}$, can we place n intervals with integer endpoints within $[0, m]$ and lengths $a_1, a_2, \ldots, a_n$, respectively, such that the number of intervals overlapping $(i, i + 1)$ is exactly the target density t_i? In other words, can we choose nonnegative integer offsets $b_1, b_2, \ldots, b_n$ such that $a_j + b_j \le m$ for each j $(1 \le j \le n)$; and, for each i $(0 \le i < m)$, there are exactly t_i indices j such that $b_j \le i < a_j + b_j$?*

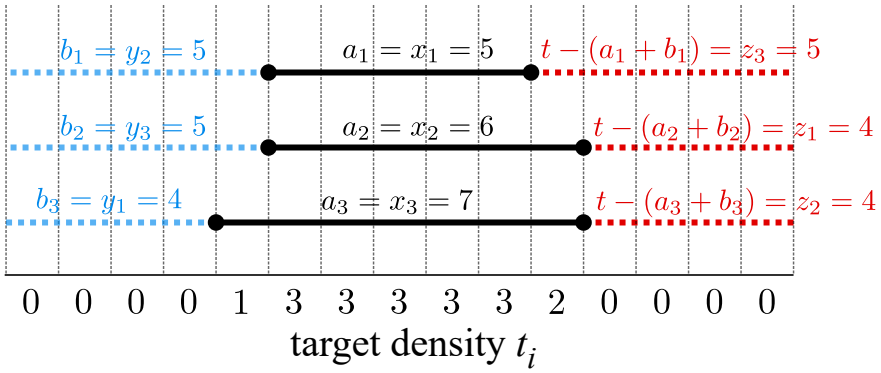

Figure 6: Length Offsets instance obtained by reducing from Numerical 3-Dimensional Matching where $X = \{5, 6, 7\}, Y = \{4, 5, 5\}, Z = \{4, 4, 5\}, t = 15$; and its solution corresponding to the Numerical 3-Dimensional Matching solution of $(5, 5, 5)$, $(5, 6, 4)$, $(4, 7, 4)$.

Theorem 3.1. Length Offsets *is parsimoniously reducible from* Numerical 3-Dimensional Matching *in which at least one of the three multisets is actually a set.*

Proof. We give a parsimonious reduction from Numerical 3-Dimensional Matching where X is a set, as guaranteed by Theorem 2.4. Specifically, consider a Numerical 3-Dimensional Matching instance with set $X = \{x_1, \ldots, x_n\}$, multisets Y and Z, and a target sum t. Assume without loss of generality that every element of X, Y, Z is strictly between $t/4$ and $t/2$.[6] We construct a Length Offsets instance that we claim has the same number of solutions: the n lengths are given simply by $a_i = x_i$, and the target densities are given by $t_i = n - |\{y \in Y : y > i\}| - |\{z \in Z : t - z \leq i\}|$, where $0 \leq i < m$ and $m = t$. See Figure 6 for an example. The intuition is that we place intervals for X, Y, Z, left-align the intervals for Y, right-align the intervals for Z, and count the remaining density for X intervals.

It remains to show that every solution to the original Numerical 3-Dimensional Matching instance corresponds to a solution to the constructed Length Offsets instance, and different Numerical 3-Dimensional Matching solutions correspond to different Length Offsets solutions. Equivalently, we will provide an injective map from Numerical 3-Dimensional Matching solutions to Length Offsets solutions, and an injective map from Length Offsets solutions to Numerical 3-Dimensional Matching solutions.

N3DM to Length Offsets. To convert a Numerical 3-Dimensional Matching solution into a Length Offsets solution, we assign $b_j = y_k$ for each solution triple (x_j, y_k, z_ℓ). Figure 6 shows this solution for the example.

For each i and each triple (x_j, y_k, z_ℓ), $b_j \leq i < a_j + b_j$ if and only if $y_k \leq i$ and $i < x_j + y_k = t - z_\ell$, so either

1. $y_k > i$; or
2. $t - z_\ell \leq i$; or

[6] If a Numerical 3-Dimensional Matching instance has any elements $\geq t$, it trivially has no solutions (as all elements are positive). Otherwise, we can convert it to an instance with this property by adding t to each element in X, Y, Z and changing the target sum from t to $\hat{t} = 4t$. Then every element is strictly between t and $2t$, and thus strictly between $\hat{t}/4 = 4t/4$ and $\hat{t}/2 = 4t/2$.

3. $b_j \leq i < a_j + b_j$.

The first case applies $|\{y \in Y : y > i\}|$ times, and the second case applies $|\{z \in Z : t - z \leq i\}|$ times, so the third case applies $n - |\{y \in Y : y > i\}| - |\{z \in Z : t - z \leq i\}| = t_i$ times. Thus our choice of the offsets is a valid solution to the LENGTH OFFSETS instance.

If two NUMERICAL 3-DIMENSIONAL MATCHING solutions differ, then (using that X is a set) some x is matched with a different y in each solution, so when converting those solutions to LENGTH OFFSETS solutions, we assign the corresponding length different offsets.

LENGTH OFFSETS to N3DM. To convert a LENGTH OFFSETS solution into a NUMERICAL 3-DIMENSIONAL MATCHING solution, for each length–offset pair (a_i, b_i), we match the triple $(a_i, b_i, t - a_i - b_i)$. These triples obviously sum to t and are therefore legal, but we need to show that their elements exist and cover X, Y, and Z, respectively.

1. Every a_i is an x_i and vice versa, so X is covered by $\{a_i\}$.

2. For each i, we have

$$\begin{aligned} t_i - t_{i-1} &= (n - |\{y \in Y : y > i\}| - |\{z \in Z : t - z \leq i\}|) \\ &\quad - (n - |\{y \in Y : y > i - 1\}| - |\{z \in Z : t - z \leq i - 1\}|) \\ &= |\{y \in Y : y = i\}| - |\{z \in Z : t - z = i\}| \end{aligned}$$

   For $i < t/2$, the second term is 0 (because $z < t/2$ by assumption), so $t_i - t_{i-1}$ is precisely the number of elements of Y that equal i. On the other hand, in a LENGTH OFFSETS instance, when $i < t/2$, $t_i - t_{i-1}$ is the number of segments which pass through i but not $i - 1$, i.e., the number of segments which begin at i and therefore have offset $b_j = i$. Therefore, Y is covered by $\{b_j\}$.

3. Following the same argument as above, but for $i > t/2$, we have that $t_i - t_{i-1} = -|\{z \in Z : t - z = i\}|$. On the other hand, in the LENGTH OFFSETS problem, for $i > t/2$, $t_i - t_{i-1}$ is the negative of the number of segments which *end* at $i - 1$; i.e., it is the negative of the number of indices j such that $a_j + b_j = i$, or equivalently, $t - (t - (a_j + b_j)) = i$. Thus, Z is covered by $\{t - (a_j + b_j)\}$ as claimed.

Different solutions to the LENGTH OFFSETS instance correspond to different solutions to the NUMERICAL 3-DIMENSIONAL MATCHING instance because, if two LENGTH OFFSETS solutions differ, then some length a_j gets different offsets, so the corresponding x_j is matched to different elements of Y in the two NUMERICAL 3-DIMENSIONAL MATCHING solutions.

We have shown two injective maps between solutions of the LENGTH OFFSETS instance and solutions of the NUMERICAL 3-DIMENSIONAL MATCHING instance, so our reduction is parsimonious. □

We now make a brief observation that we make use of later.

Lemma 3.2. *In every solution to* LENGTH OFFSETS *instances produced by the reduction in the proof of Theorem 3.1, no line segment shares its left endpoint with the right endpoint of another.*

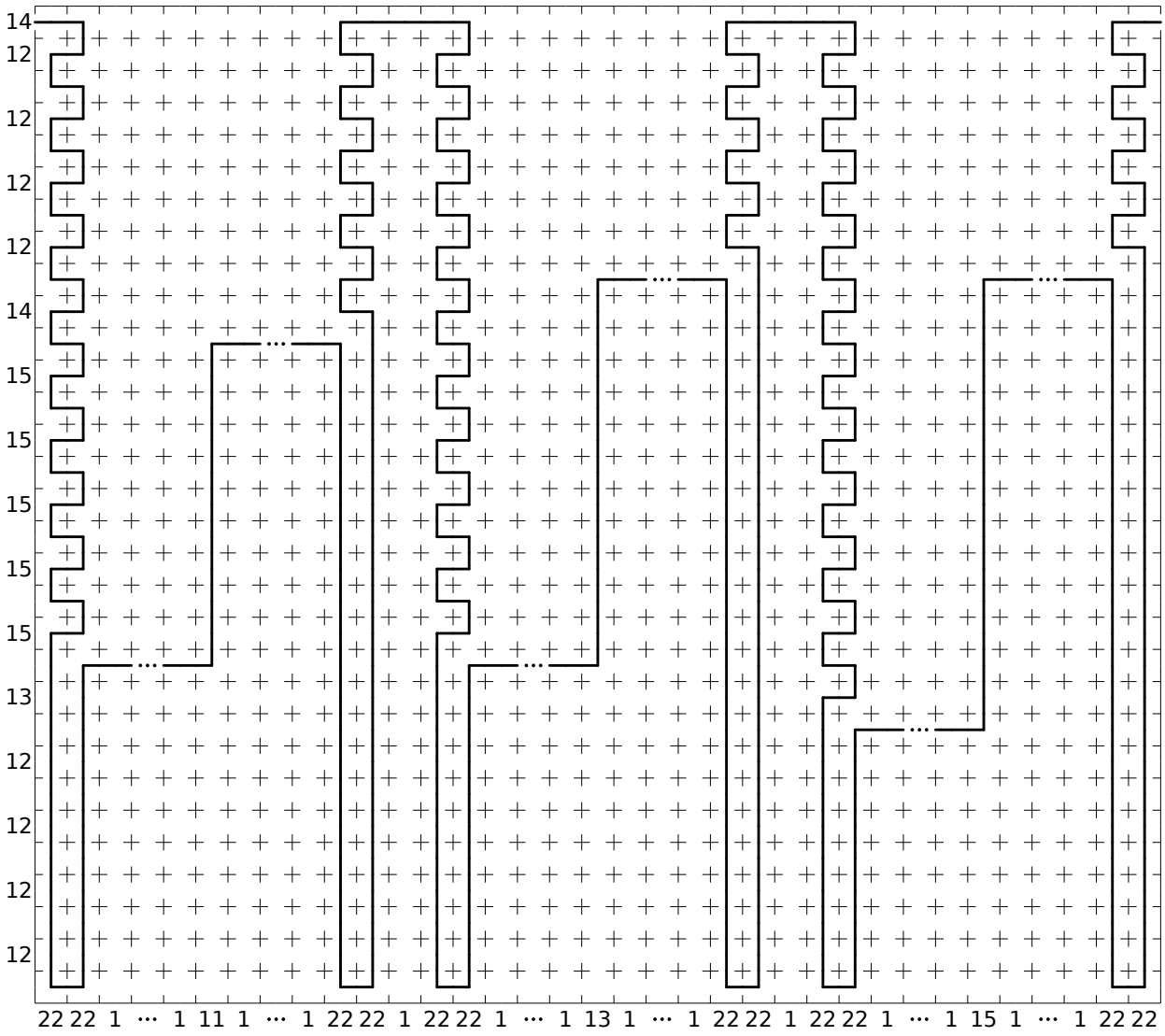

Figure 7: Solution to a Path Puzzle instance, reduced from Length Offsets from Figure 6 where $n = 3, m = 15, a_i = (5, 6, 7), t_i = (0, 0, 0, 0, 1, 3, 3, 3, 3, 3, 2, 0, 0, 0, 0)$. Ellipses elide sections of $6n = 18$ columns each labeled 1.

Proof. By assumption, all elements of the Numerical 3-Dimensional Matching instance lie in the exclusive interval $(t/4, t/2)$. Our reduction sets $a_i = x_i$, so $t/4 < a_i < t/2$. Our mapping between solutions assigns $b_i = y_k$, so $t/4 < b_i < t/2$. Adding these inequalities yields $t/2 < a_i + b_i < t$ for all i. Then $b_i < t/2 < a_j + b_j$ for all i and j, so the sets of left and right endpoints are disjoint. □

We are now ready to prove our main theorem.

Theorem 3.3. Path Puzzle *is NP-, #P- and ASP-hard.*

Proof. We give a parsimonious reduction from Length Offsets, as produced by Theorem 3.1 so

that Lemma 3.2 applies. Given a LENGTH OFFSETS problem with lengths $a_1, \ldots, a_n$ and target densities $t_0, \ldots, t_{m-1}$, we construct an equivalent PATH PUZZLE instance as follows. Figure 7 shows our construction instantiated for the same LENGTH OFFSETS instance from Figure 6.

Dimensions: The grid has $2m + 3$ rows and $(12n + 6)n - 1$ columns.

We group the columns into n *blocks* $B_1, \ldots, B_n$ of $12n + 5$ columns each, interspersed with $n - 1$ *lone columns*. Thus block B_i $(1 \le i \le n)$ consists of columns $(12n + 6)i - (12n + 5), \ldots, (12n + 6)i - 1$ and the ith lone column $(1 \le i < n)$ is column $(12n + 6)i$.

Doors: We place doors at the left and right ends of the top row.

Row labels: Counting up from the bottom, the $(2i + 2)$nd row has a label of $4n + t_i$, for each i $(0 \le i \le m)$; the $(2m + 2)$nd and $(2m + 3)$rd (topmost) rows have labels $4n$ and $5n - 1$, respectively; and all other row labels are blank.

Column labels: Each lone column has a column label of 1. Each block B_j $(1 \le j \le n)$ has labels of $2m + 3$ on its first two and last two columns; a label of $2a_j + 1$ on its middle column ($6n + 3$th column); and labels of 1 on all other columns (which split into two sections of $6n$ consecutive columns).

Any solution to this path puzzle has the following properties:

1. Every square in the first two and last two columns of each block is visited, by the column labels of $2m + 3$.

2. Every section of $6n$ consecutive columns labeled 1 (within a block) corresponds to a single horizontal path, which must be in one of the blank rows because all row labels are less than $6n$.

3. Every column labeled $2a_i + 1$ is a single vertical line segment, because both neighboring columns are labeled 1, just enough to enter and exit the column once.

4. No top square of a column labeled $2a_i + 1$ is visited, because if one were, the second square from the top of such a column would also be visited (by Property 3 and because $2a_i + 1 > 1$), but then that row would have more than $4n$ visited squares (by Property 1).

5. In each line column, the top square (and only that square) is visited, because those are the remaining squares on the top row that can be visited, and are just enough (with Property 1) to account for a total of $5n$.

6. The vertical line segment in the (unique) column labeled $2a_i + 1$ (from Property 3) visits a_i rows with labels of $4n + t_j$ for various j. Of that $4n + t_j$, $4n$ visits are accounted for by the full columns of Property 1, so the positions of those line segments are a solution to the LENGTH OFFSETS problem.

7. Given placements of the vertical line segments corresponding to a valid solution to the LENGTH OFFSETS problem, we claim that the rest of the path is uniquely determined. The set of visited squares in each gadget is uniquely determined by the previous properties. In each pair of columns labeled $2m + 3$:

(a) The bottom two squares each have only two visited neighbors, each other and the square above them, so each of them connects by the path to those two squares.

(b) For squares in the pair of columns below the entry point of the length $6n$ horizontal path, the long U-shaped path shown in Figure 7 is forced. For each horizontal pair of squares except the bottom pair, the squares below connect to them. If the pair squares connect to each other, they form a closed loop, so they must connect to the squares above them instead.

(c) For squares above the entry point of the length $6n$ horizontal path, the zig-zag path is forced. The pair of columns divides evenly into 2×2 chunks because the entry point of the length $6n$ path is in a row with no label, and the only such rows are at even height. Let *inside* and *outside* be relative to the center of the block. In each chunk, the bottom inside square can't connect to the square below (because that square is already known to connect down and to the inside), so it connects to the outside and up. Similarly, the top outside square can't connect to the square below (because that square is already known to connect down and to the bottom inside square), so it connects to the inside and up (except that in the very top 2×2 chunk, the top outside square can't connect down and can and must connect to the outside to satisfy the top row).

Thus, each solution to the path puzzle determines a solution to the Length Offsets problem, and that solution is uniquely determined, so the number of solutions to the Length Offsets problem is the same as the number of solutions of the path puzzle, and the reduction is parsimonious as desired. Note that we are relying on the uniqueness of the lengths a_i from the Length Offsets problem definition; otherwise, permuting which copy of a duplicated length gets which offset in the path puzzle would generate multiple solutions to Path Puzzle from each solution of Length Offsets. □

In fact, our reduction can be converted into one giving complete information (i.e., all row and column labels), demonstrating that partial information is not the source of Path Puzzle's hardness.

Theorem 3.4. *Perfect-information* Path Puzzle *(with all row and column labels given as labels) is NP-, #P- and ASP-hard.*

Proof. Recall that the reduction from the proof of Theorem 3.3 (referring to Figure 7) already provides all column sums and about half of the row sums. We show how to provide the remaining row sums without giving away information about the solution to the original Length Offsets instance.

The rows with missing labels are the $(2i-1)$st rows for $i = 1, 2, \ldots, m$. In each such row, the solution path must visit $(6n+2)r_i$ cells, where r_i is the number of segments in the Length Offsets solution which have an endpoint at i. Recall from Lemma 3.2 that no line segment shares its left endpoint with the right endpoint of another. Thus there is only one type of endpoint at each coordinate i, and we can compute $r_i = |t_{i+1} - t_i|$. The value of r_i depends only on the Length Offsets instance, not its solutions, so we can modify our reduction to specify a label of $(6n+2)r_i$ for row $2i-1$, producing a perfect-information instance of Path Puzzle. □

4 Open Problems

One interesting open problem is whether Planar 3DM, where the bipartite graph of elements and triples is planar, is also ASP-hard and #P-hard. This problem is known to be NP-hard [DF86], and the variable–clause gadget structure in the proof of Theorem 2.2 is close to preserving planarity. Unfortunately, the initial clause tripling destroys any planarity in the input, and seems difficult to avoid.

Another intriguing open problem is whether discrete tomography with partial information, but no Hamiltonian path constraint, is NP-hard. If true, this would be another aspect of path puzzles which make them hard.

Acknowledgements

We thank Jayson Lynch for useful discussions and debugging help, and Quanquan Liu for help in constructing the figures for this paper. Most figures were produced using SVG Tiler (https://github.com/edemaine/svgtiler).

References

[Ant14] Gary Antonick. Roderick Kimball's path puzzles. *The New York Times: Numberplay*, 28 July 2014. https://wordplay.blogs.nytimes.com/2014/07/28/path-2/.

[DF86] M. E. Dyer and A. M. Frieze. Planar 3DM is NP-complete. *Journal of Algorithms*, 7(2):174–184, 1986.

[DLN12] Alberto Del Lungo and Maurice Navat. Reconstruction of connected sets from two projections. In Herman and Kuba [HK12], chapter 7.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[HK12] Gabor T. Herman and Attila Kuba, editors. *Discrete Tomography: Foundations, algorithms, and applications*. Birkhäuser, 2012.

[HMRS98] Harry B. Hunt, III, Madhav V. Marathe, Venkatesh Radhakrishnan, and Richard E. Stearns. The complexity of planar counting problems. *SIAM Journal on Computing*, 27(4):1142–1167, 1998.

[Kim13] Roderick Kimball. *Path Puzzles*. Roderick Kimball, 2nd edition, 10 2013.

[Set02] Takahiro Seta. The complexities of puzzles, Cross Sum, and their Another Solution Problems (ASP). Senior thesis, University of Tokyo, 2002.

[Val79] Leslie G Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

A Solution to the Font Puzzles

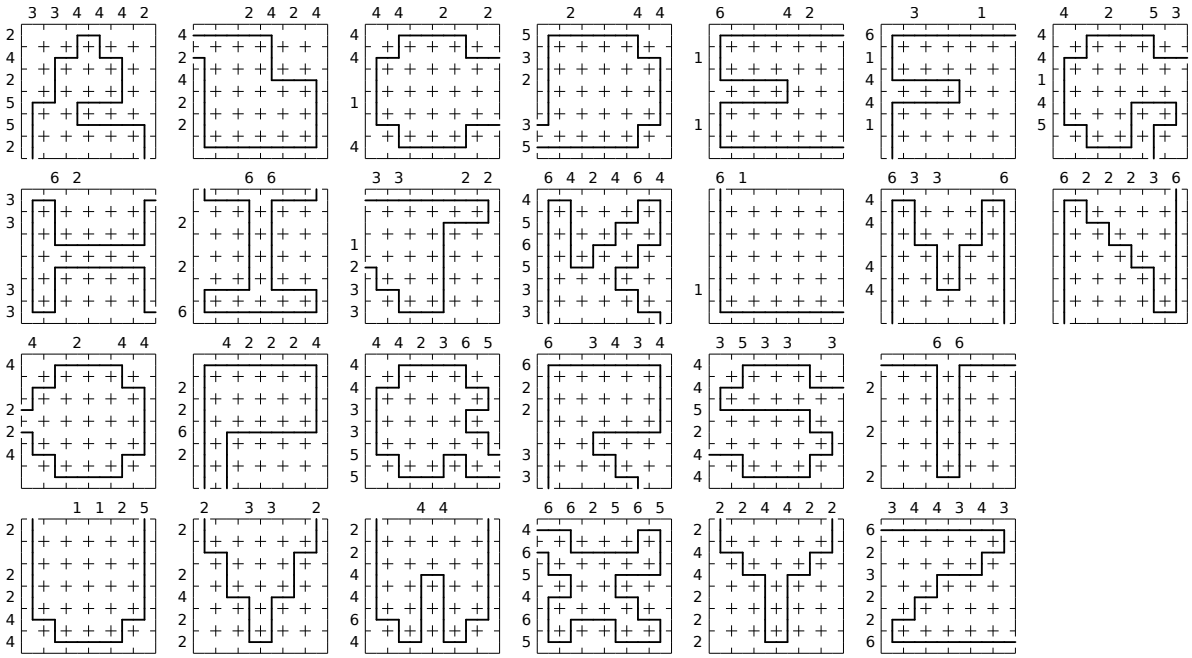

Figure 8: Solved font