Generating Constrained Lattice Paths in a Grid Related to Counting Cycles

Hareshkumar Jadav[1,*,†], Mihir Patel[1,†], Samip Shah[1,†], Ranveer Singh[1,†] and Harsh Talati[1,†]

[1]Indian Institute of Technology Indore, Khandwa Rd, Simrol, Madhya Pradesh , India 453552

Abstract

This paper gives a bijection between two classes of lattice paths in an $n \times n$ grid. This bijection is crucial for identifying short cycles in simple bipartite graphs. We also give an algorithm for generating such paths with the complexity of $\mathcal{O}\left(\frac{4^n}{n^{1/2}}\right)$. The proposed work lays the groundwork for counting cycles of length from g to $2g-2$ in bipartite graphs, where g is the girth of the graph. which has broader implications on Error correcting codes, cryptography etc.

Keywords

Bipartite Graphs, Cycle Counting, Lattice Paths, Algorithms

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

1. Introduction

The computational challenge of counting cycles in graph structures represents a persistent research frontier, particularly given its #P-complete problem. This complexity has driven specialized investigations into graph classes with practical applications. Among these, bipartite graphs hold significant importance due to their role in communication systems, where they form Tanner graphs for error-correcting codes like LDPC codes [1, 2]. There are works in counting short cycles in a bipartite graph [3, 4, 5, 6].

Recently, in 2020, Dehghan and Banihashem [3] proposed an algorithm to calculate the multiplicity of short cycles of lengths g, $g+2$, and $g+4$, where g denotes the girth of an undirected bipartite graph. They used a combinatorial approach to design an algorithm for enumerating short cycles. In this approach, they identified specific patterns within a modified BFS tree, as illustrated in Figure 1 and Figure 2, corresponding to cycles of lengths $g+2$ and $g+4$, respectively. Building on this, our work extends the idea of pattern generation in bipartite graphs, aiming to develop a more general algorithm to count cycles of lengths g, $g+2$, up to $2g-2$ in bipartite graphs. These patterns can be represented as paths in an $n \times n$ grid, from $(0,0)$ to (n,n), as shown in Figure 3.

We map these patterns as lattice paths in the $n \times n$ grid. The set of all integer vectors in the d-dimensional space can be written as: $\mathbb{Z}^d = \{(x_1, x_2, \ldots, x_d) \mid x_i \in \mathbb{Z}\}$. A *lattice path* K in $\mathbb{Z}^d$ of length k is a sequence $(v_0, v_1, \ldots, v_k)$, where $v_i \in \mathbb{Z}^d$, such that each consecutive difference $v_i - v_{i-1}$ lies in $, where $ is the set of step vectors [7]. $L_m(a \to e; |R)$ represents the set of all lattice paths from point a to e that follows a defined restriction R while taking steps that belong to $ [8].

In this paper, we focus on counting and enumerating grid paths (2-dimensional lattice paths) of length $2n$ that start at $a = (0,0)$ and end at $e = (n,n)$, using the set of steps $ = \{(0,1),(1,0)\}$. For simplicity,

ICTCS 2025: Italian Conference on Theoretical Computer Science, September 10–12, 2025, Pescara, Italy

*Corresponding author.

†These authors contributed equally.

phd2301106001@iiti.ac.in (H. Jadav); cse210001050@alum.iiti.ac.in (M. Patel); cse210001061@alum.iiti.ac.in (S. Shah); ranveer@iiti.ac.in (R. Singh); cse210001022@alum.iiti.ac.in (H. Talati)

https://ranveeriit.github.io (R. Singh)

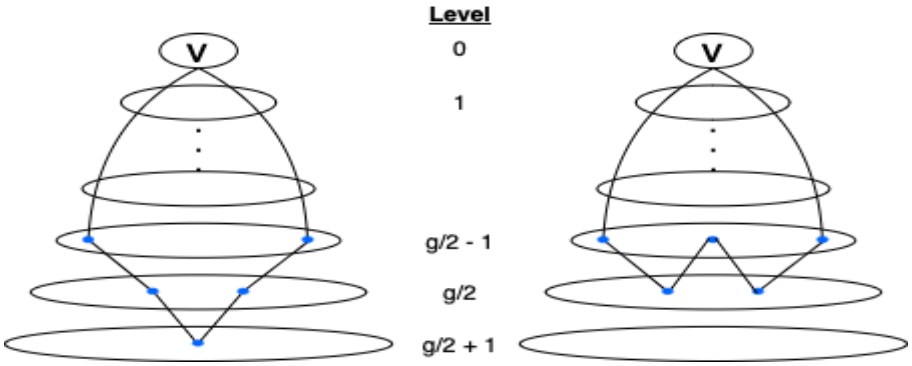


Figure 1: The BFS tree and the corresponding patterns for cycles of length $g + 2$, as presented in [3].

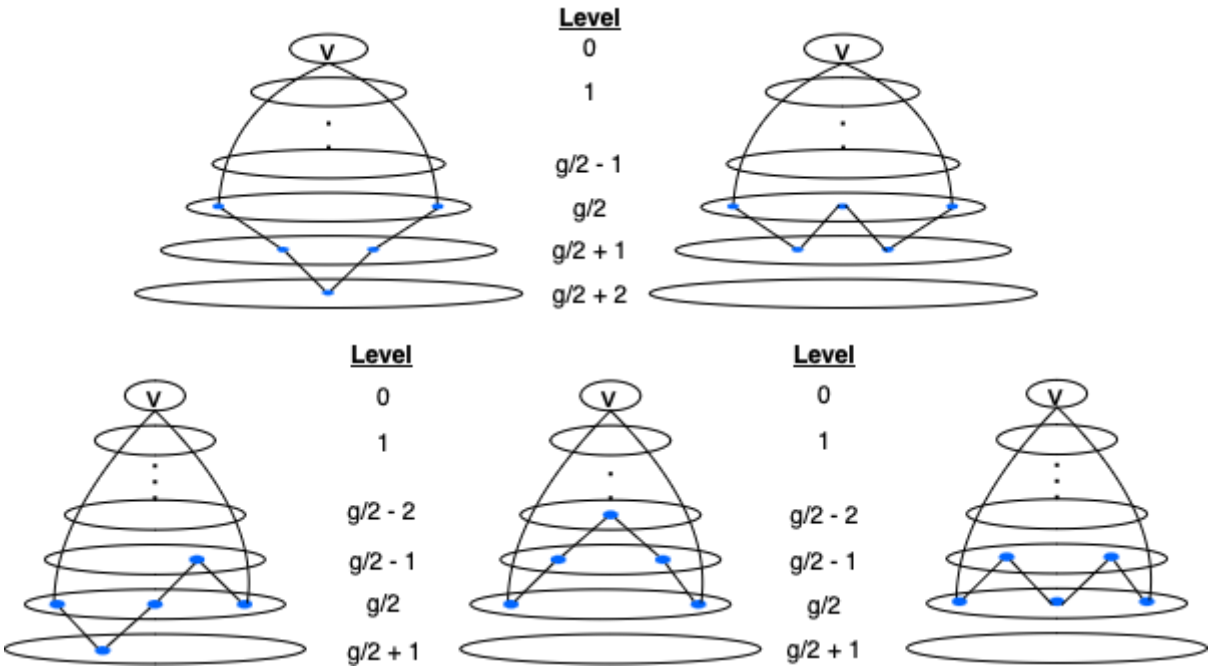


Figure 2: The BFS tree and the corresponding patterns for cycles of length $g + 4$, as presented in [3].

we omit \$ from the notation of L. The set of such paths subject to given restrictions R is denoted by $L_{2n}(a \to e \mid R)$.

2. Defining the Special Constrained Path

In this section, we define special lattice paths motivated by the patterns described in [3]. The motivation for such patterns is that some are impossible in the modified BFS tree. We denote the i^{th} level of this modified BFS tree by l_i. For example, in Figure 5, the two vertices v_0 and v_1 at level $l_{\frac{g}{2}-3}$ have a length path $\frac{g}{2} - 3$ from the source v. Thus, the cycle $v - v_0 - v_1 - v$ has length $2(\frac{g}{2} - 3) + 2 = g - 4 < g$, which contradicts the girth condition. Therefore, the path is invalid.

This translates to a restriction we call *Right-Then-Up*: a path cannot go down and then up above the line $l_{\frac{g}{2}}$ (see Figure 5). Since our work rotates this grid (see Figure 3), our path cannot go "Right-Then-Up" above the line $y = x$. We will define this formally in Definition 3.

Definition 1 (Lattice Paths). *We use the notation $L_m(a \to e \mid R)$ to denote all lattice paths from point a to point e with restrictions R. If the starting point a, the ending point e, and the length m are not explicitly*

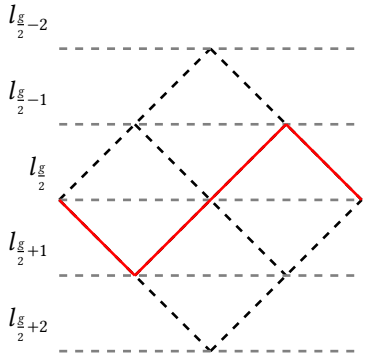


Figure 3: Equivalent representation of the Type 3 pattern shown in Figure 2 in a 2×2 grid.

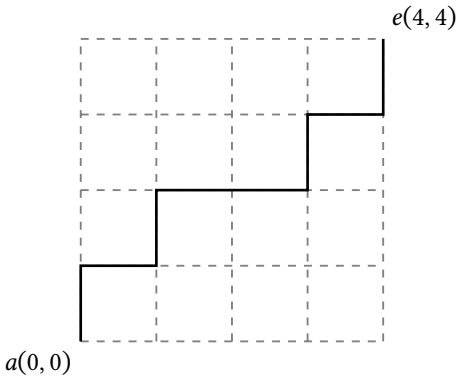


Figure 4: Lattice path in 4×4 grid

provided, they are assumed to be $(0,0)$, (n,n), *and* $m = 2n$, *respectively.*

$$L(R) = L_{2n}((0,0) \to (n,n) \mid R)$$

Definition 2 ($x+2$ Constrained Paths)**.** *An $x+2$ constrained path, as illustrated in Figure 6, is a lattice path such that $x_i + 2 \geq y_i$ for all points $v_i = (x_i, y_i)$ along the path. This constraint ensures that the path stays on or below the line $y = x + 2$ throughout its traversal. The set of all such paths is denoted by:*

$$\mathbb{P} = L(\, x_i + 2 \geq y_i \;\; \forall i \in \{0, 1, \ldots, 2n\})$$

Definition 3 (Right-Then-Up Constrained Paths)**.** *For every consecutive pair of steps, if the step from v_{i-1} to v_i is a right step $(1,0)$, and $v_i = (x_i, y_i)$ satisfies $y_i \geq x_i + 1$, then the next step from v_i to v_{i+1} must also be a right step $(1,0)$. The set of Right-Then-Up Constrained Paths is denoted by:*

$$\mathbb{H} = L\,(\, v_i - v_{i-1} = (1,0) \text{ and } y_i \geq x_i + 1 \Rightarrow v_{i+1} - v_i = (1,0),\;\; \forall i \in \{1, 2, \ldots, 2n-1\}) \tag{1}$$

It can be observed that these paths form a specific "inverted L-shaped" structure above the line $y = x$, as illustrated in Figure 7a.

3. Bijection between Paths

In this section, we show that there exists a bijection between the $x+2$ constrained paths ($\mathbb{P}$) and the Right-Then-Up constrained paths ($\mathbb{H}$). Before this, we first define a few notations and a supporting lemma.

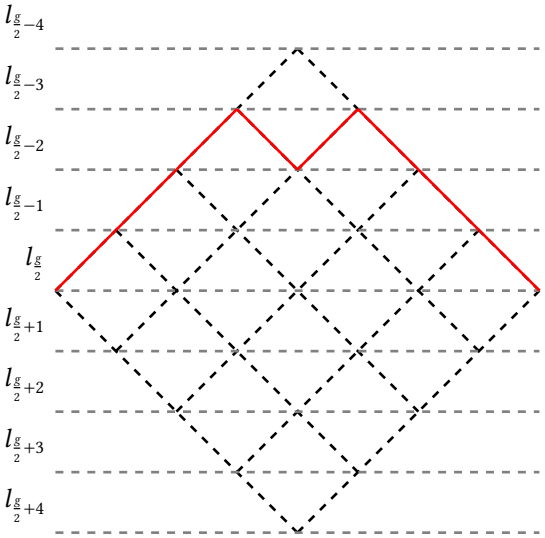


Figure 5: This pattern represents a possible cycle of length $g + 8$ in the modified BFS tree. However, such a cycle is invalid or does not exist because there is a vertex located above level $l_{g/2}$ that has two connected vertices, which could form a cycle shorter than the girth g. This contradiction renders the pattern invalid, refer [3]. Consequently, this condition is used as a constraint for Right-Then-Up Constrained paths.

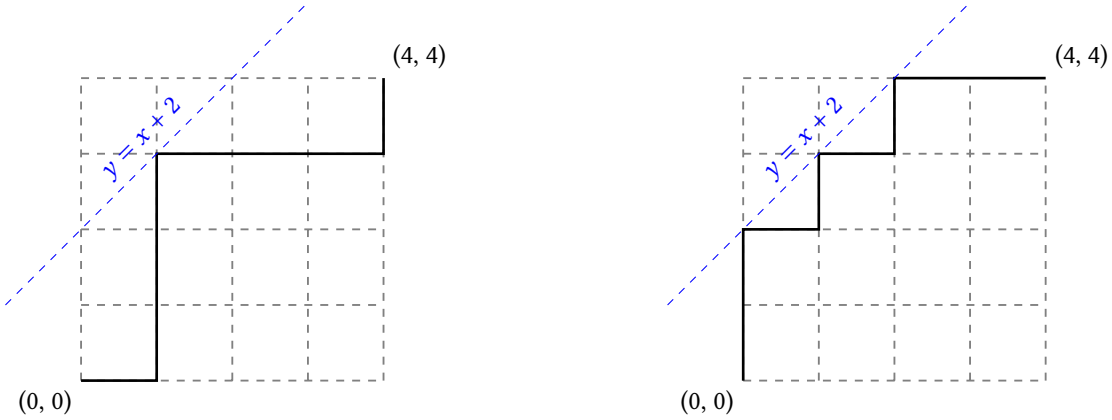


Figure 6: Examples of $x + 2$ *constrained paths*

Definition 4. *A subpath of a lattice path* $K = (v_0, v_1, \dots, v_{2n})$ *is a contiguous subset of points in* K*, denoted by* $SP_K(a, b)$*, where* a *and* b *are the indices of the endpoints:*

$$SP_K(a, b) = (v_a, v_{a+1}, \dots, v_b).$$

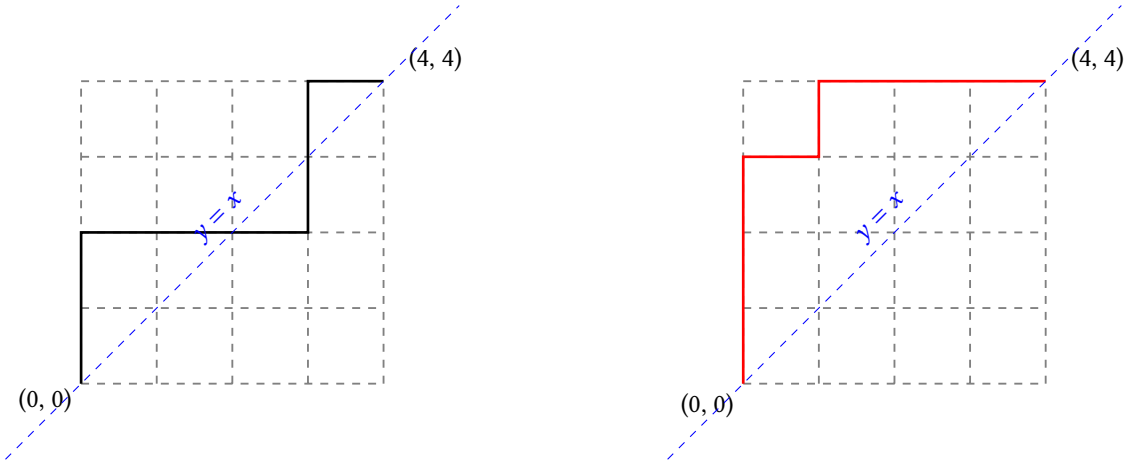
Definition 5. *We define a function* $T : \mathbb{H} \to \mathbb{P}$ *such that, for each lattice point in the path* $K = (v_0, v_1, \dots, v_{2n}) \in \mathbb{H}$*, we apply a function* f*:*

$$T(K) = (f(v_0), f(v_1), \dots, f(v_{2n})),$$

where $f : \mathbb{Z}^2 \to \mathbb{Z}^2$ *is defined for a point* $v = (x, y)$ *as*

$$f(v) = \begin{cases} v & \text{if } x + 2 > y, \\ \left(\left\lfloor \frac{x+y-1}{2} \right\rfloor, \left\lfloor \frac{x+y+2}{2} \right\rfloor\right) & \text{if } x + 2 \leq y. \end{cases}$$

Alternatively, this function f can be defined in terms of the index of the point. Since K starts at $(0, 0)$ and consists of only single steps in either direction, we can observe that for any point $v_i = (x_i, y_i)$ in K,



(a) Valid path (b) Invalid path

Figure 7: Comparison of valid and invalid *Right-Then-Up constrained paths.*

we have $i = x_i + y_i$. Thus, we can rewrite f as follows:

$$f(v_i) = \begin{cases} v_i & \text{if } x_i + 2 > y_i, \\ \left(\left\lfloor \frac{i-1}{2} \right\rfloor, \left\lfloor \frac{i+2}{2} \right\rfloor\right) & \text{if } x_i + 2 \leq y_i. \end{cases}$$

We can easily verify that $f(v_i)$ lies on or below the line $y = x + 2$, thus $T(K) \in \mathbb{P}$. From this point onward, we denote v_i as the i^{th} point in $K \in \mathbb{H}$.

Lemma 1. *If v_i lies on or above the line $y = x + 2$ (i.e., $y_i \geq x_i + 2$), then $f(v_i)$ will lie either on the line $y = x + 1$ or $y = x + 2$.*

Proof. If i is odd, let $i = 2m + 1$. Then $f(v_i) = (m, m + 1)$, which lies on the line $y = x + 1$. If i is even, let $i = 2m$. Then $f(v_i) = (m - 1, m + 1)$, which lies on the line $y = x + 2$. Hence, in both cases, $f(v_i)$ lies either on $y = x + 1$ or $y = x + 2$. □

Lemma 2. *If $f(v_i)$ lies on or below the line $y = x$, then $f(v_i) = v_i$*

Proof. This follows from Definition 5 □

Theorem 1. *A bijection exists between $\mathbb{H}$ and $\mathbb{P}$, given by T.*

Proof. First, let us prove the injectivity of the function T. Let $K_1, K_2 \in \mathbb{H}$ such that $T(K_1) = T(K_2)$. Let v_i^1 and v_i^2 denote the i^{th} points in K_1 and K_2, respectively.

Consider any point $v_j^1 \in K_1$ such that it lies on the line $y = x + 1$. If both v_{j-1}^1 and v_{j+1}^1 lie on the line $y = x + 2$, then we have $v_j^1 - v_{j-1}^1 = (1, 0)$ (i.e., the path moved to the right), and $v_{j+1}^1 - v_j^1 = (0, 1)$ (i.e., the path moved up). Since $y_j = x_j + 1$, this contradicts the restrictions of $\mathbb{H}$. This implies that at least one of v_{j-1}^1 or v_{j+1}^1 must lie on the line $y = x$. Denote such a point by v_k^1.

As v_k^1 lies below the line $y = x + 2$, we know that $f(v_k^1) = v_k^1$. Since $f(v_k^1) = f(v_k^2)$, it follows that $v_k^1 = f(v_k^2)$, which means $f(v_k^2)$ lies on the line $y = x$. From Lemma 2, we conclude that $f(v_k^2) = v_k^2$. The last two equations imply that $v_k^1 = v_k^2$. Therefore, v_k^2 also lies on the line $y = x$. Now, since v_j^2 is adjacent to v_k^2, it must lie below the line $y = x + 2$. This implies $f(v_j^2) = v_j^2$. Given that $f(v_j^1) = f(v_j^2)$, and v_j^1 lies on the line $y = x + 1$, it follows that $f(v_j^1) = v_j^1$, hence $v_j^1 = v_j^2$. Thus, we have shown that the points on the line $y = x + 1$ are the same for both K_1 and K_2.

Let the indices of points that lie on the line $y = x + 1$ be $a_0, a_1, a_2, \ldots, a_s$. Consider the subpaths $SP_{K_1}(a_i, a_{i+1})$ and $SP_{K_2}(a_i, a_{i+1})$. To show that $K_1 = K_2$, it is sufficient to prove that

$$\forall i \in \{0, 1, \ldots, s - 1\}, \quad SP_{K_1}(a_i, a_{i+1}) = SP_{K_2}(a_i, a_{i+1}),$$

along with

$$SP_{K_1}(0, a_0) = SP_{K_2}(0, a_0) \quad \text{and} \quad SP_{K_1}(a_s, 2n) = SP_{K_2}(a_s, 2n).$$

This would complete the proof of the injectivity of T.

We note that the subpaths mentioned above cannot cross or touch the line $y = x + 1$; they must lie entirely either above or below this line, except at the endpoints. Let us denote one such pair of subpaths as $SP_{K_1}(a, b)$ and $SP_{K_2}(a, b)$. We first prove that these subpaths lie on the same side of the line $y = x + 1$.

Assume, for the sake of contradiction, that $SP_{K_1}(a, b)$ and $SP_{K_2}(a, b)$ lie on opposite sides of the line $y = x + 1$, and without loss of generality, suppose $SP_{K_1}(a, b)$ lies above the line. Then, for all $j \in \{a, a + 1, \dots, b\}$, we have $y_j^1 > x_j^1 + 1$ and $y_j^2 < x_j^2 + 1$. Since $y_j^1 \geq x_j^1 + 2$, we use the definition:

$$f(v_j^1) = \left(\left\lfloor \frac{j-1}{2} \right\rfloor, \left\lfloor \frac{j+2}{2} \right\rfloor \right).$$

This means $f(v_j^1)$ lies above the line $y = x + 1$. On the other hand, $f(v_j^2) = v_j^2$ lies below the line $y = x + 1$, which contradicts the assumption that $T(K_1) = T(K_2)$. Hence, both subpaths must lie on the same side of the line $y = x + 1$. Next, we show that all the points in the corresponding subpaths of K_1 and K_2 coincide.

- *First case:* Both subpaths lie above the line $y = x + 1$. There exists only one unique inverted L-shaped path[1] from v_a to v_b that satisfies the constraints of $\mathbb{H}$. Hence,

$$SP_{K_1}(a, b) = SP_{K_2}(a, b).$$

- *Second case:* Both subpaths lie below the line $y = x + 1$. Since $f(v) = v$ for points below the line $y = x + 2$, we have:

$$T(SP_{K_1}(a, b)) = SP_{K_1}(a, b) \quad \text{and} \quad T(SP_{K_2}(a, b)) = SP_{K_2}(a, b).$$

  Given that $T(K_1) = T(K_2)$, it follows that:

$$T(SP_{K_1}(a, b)) = T(SP_{K_2}(a, b)) \implies SP_{K_1}(a, b) = SP_{K_2}(a, b).$$

Therefore, T is injective.

Now, let us prove the surjectivity of the function T. We aim to show that for any path $K_1 \in \mathbb{P}$ (a path constrained below the line $y = x + 2$), there exists a path $K_2 \in \mathbb{H}$ (a Right-Then-Up constrained path) such that $T(K_2) = K_1$. We will prove this by explicitly constructing K_2 from K_1.

Let K_1 be a lattice path in $\mathbb{P}$. Suppose $c_1, c_2, \dots, c_s$ are the indices of the points on K_1 that lie on the line $y = x + 1$, where the point immediately before lies on $y = x$. Similarly, let $d_1, d_2, \dots, d_s$ be the indices of the points on K_1 that lie on $y = x + 1$, where the point immediately after lies on $y = x$. In Figure 8a, the first set of indices (the c-values) indicates "entry" into the region $y \geq x + 1$ (blue lines), while the second set (the d-values) indicates "exit" from this region (orange lines).

As the path starts from $(0, 0)$, which lies outside the region $y \geq x + 1$, it must first enter the region before it can exit. Therefore, the following inequality holds:

$$0 < c_1 \leq d_1 < c_2 \leq d_2 < \cdots < c_s \leq d_s < 2n. \tag{2}$$

We will refer to the set of corresponding subpaths determined by the indices in the above equation as the *zig-zag partition*, denoted by ZP, because one can observe zig-zag paths between the lines $y = x + 1$ and $y = x + 2$, as illustrated in Figure 8a.

[1]For any subpath $SP_K(a, b)$ lying above $y = x + 1$, it must make $R = x_b - x_a$ right steps and $U = y_a - y_b$ up steps. The restrictions of $\mathbb{H}$ do not allow an up step after a right step. Therefore, the subpath must make all U up steps first, followed by all R right steps, forming a unique inverted L-shaped path.

For each subpath of K_1 corresponding to the "zig-zag partition," we construct the corresponding subpath of K_2. After the construction, to prove that $T(K_2) = K_1$, it will be sufficient to show that for all subpaths $SP_{K_1}(a, b) \in ZP$, we have

$$T(SP_{K_2}(a, b)) = SP_{K_1}(a, b).$$

This will establish that T is surjective. We observe that, due to the way we defined the partition, each subpath is either entirely on or above the line $y = x + 1$ (i.e., inside the region $y \geq x + 1$), or entirely on or below the line $y = x + 1$ (i.e., outside the region $y \geq x + 1$).

We will construct the subpaths of K_2 using the following two cases.

- *First case:* For subpaths below the line $y = x+1$ in the "zig-zag partition," we retain these subpaths in K_2 as they are. That is, for all $SP_{K_1}(a, b) \in ZP$ that lie below $y = x + 1$, we set

  $$SP_{K_2}(a, b) = SP_{K_1}(a, b).$$

  Since the subpath $SP_{K_2}(a, b)$ lies entirely below $y = x + 1$, it follows from the definition of f that

  $$T(SP_{K_2}(a, b)) = SP_{K_2}(a, b) = SP_{K_1}(a, b).$$

- *Second case:* For each subpath $SP_{K_1}(a, b) \in ZP$ that lies on or above the line $y = x+1$, the endpoints will always be on the line $y = x + 1$. In this case, we construct a "reverse L-shaped" subpath in K_2, as shown in Figure 8b in teal. Specifically, $SP_{K_2}(a, b)$ is given by:

  $$SP_{K_2}(a, b) = \left(\frac{a+1}{2}, \frac{a-1}{2}\right), \left(\frac{a+1}{2}, \frac{a-1}{2} + 1\right), \ldots, \left(\frac{a+1}{2}, \frac{b-1}{2}\right), \left(\frac{a+1}{2} + 1, \frac{b-1}{2}\right), \ldots, \left(\frac{b+1}{2}, \frac{b-1}{2}\right).$$

  This subpath lies entirely above the line $y = x + 1$ and forms a valid Right-Then-Up path, as required by the constraints of $\mathbb{H}$.
  We now verify that $T(SP_{K_2}(a, b)) = SP_{K_1}(a, b)$ for such subpaths. Note that for any point v on or above the line $y = x + 1$, the image $f(v)$ also lies on or above the line $y = x + 1$. Thus, both $T(SP_{K_2}(a, b))$ and $SP_{K_1}(a, b)$ lie entirely above $y = x + 1$.
  However, since both must also lie below or on the line $y = x + 2$, and because there is a unique subpath between two endpoints v_a and v_b such that all intermediate points satisfy $x+1 \leq y \leq x+2$[2], we conclude that

  $$T(SP_{K_2}(a, b)) = SP_{K_1}(a, b).$$

For any path $K_1 \in \mathbb{P}$, we have explicitly constructed a path $K_2 \in \mathbb{H}$ such that $T(K_2) = K_1$. Thus, T is surjective. □

4. Cardinality of Right-Then-Up Constrained Paths

In the previous section, we established a bijection between $x + 2$ *constrained paths* and *Right-Then-Up constrained paths*. Thus, to compute the cardinality of Right-Then-Up constrained paths, it suffices to determine the cardinality of $x + 2$ constrained paths.

We apply a coordinate transformation from (x, y) to a new coordinate system (x', y') such that:

$$x' = x + 2 \quad \text{and} \quad y' = y.$$

Under this transformation, the number of paths

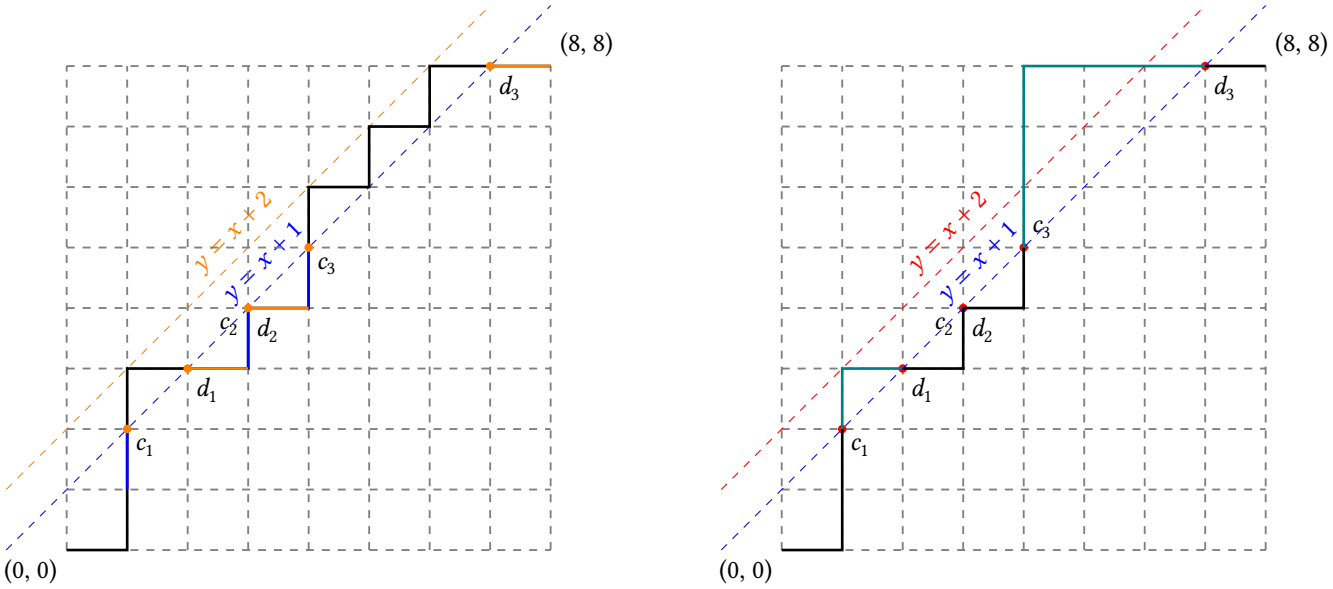
$$\left|L_{2n}\big((0, 0) \to (n, n) \,\big|\, x_i + 2 \geq y_i,\ \forall i \in \{0, 1, \ldots, 2n\}\big)\right|$$

is equivalent to

$$\left|L_{2n}\big((2, 0) \to (n + 2, n) \,\big|\, x'_i \geq y'_i,\ \forall i \in \{0, 1, \ldots, 2n\}\big)\right|.$$

We will use the following theorem for counting such paths:

[2]If a point lies on $y = x + 1$, then the next step must be an up-step; if a point lies on $y = x + 2$, then the next step must be a right-step. This enforces a unique "zig-zag" structure.



(a) Illustration of Partition points (b) Construction of K_2

Figure 8: (a) Illustration of Partition points (b) Construction of K_2

Theorem 2 ([8]). *Let $r \geq p$ and $s \geq q$. The number of all lattice paths from (p, q) to (r, s) that stay weakly below the line $y = x$ is given by:*

$$\left|L\big((p,q) \to (r,s) \mid x \geq y\big)\right| = \binom{r+s-p-q}{r-p} - \binom{r+s-p-q}{r-q+1}.$$

Applying this theorem to

$$\left|L_{2n}(v_0 \to v_{2n} \mid x_i \geq y_i, \forall i \in \{0, 1, \ldots, 2n\})\right|,$$

with $v_0 = (2, 0)$ and $v_{2n} = (n + 2, n)$, we obtain:

$$|\mathbb{P}| = |\mathbb{H}| = \left|L\big((2,0) \to (n+2,n) \mid x \geq y\big)\right| = \binom{2n}{n} - \binom{2n}{n-3}. \tag{3}$$

5. Algorithm for Generating Right-Then-Up Constrained Paths

In this section we will propose an algorithm to generate Right-Then-Up constrained path in an $n \times n$ grid. The core idea of the algorithm is to generate paths that are entirely below $y = x + 2$ and then apply the construction mentioned in the surjuctive proof above. Theorem 1 will ensure that we will generate all paths with one-to-one correspondence. Algorithm 4 generates paths that are below $y = x + 2$. While Algorithm 3 using Algorithm 1,2 transform those paths to *Right-Then-Up constrained paths.*

Algorithm 1 computes the *zigzag partition* of a given path by identifying specific points within the input path that satisfy the condition mentioned for indices in Equation 2. If a point satisfies these criteria, its index i is added to an array *partitionPoints*, which contains the indices of all partition points. These points divide the path into distinct segments represented by indices $c_1 \leq d_1 < c_2 \leq d_2 < \cdots < c_s \leq d_s$, where $[c_i, d_i]$ denotes the boundaries of the partitions. The path within each alternate interval exhibits a zigzag structure, hence the term *zigzag partitions.* Algorithm 2 transforms each "zigzag partition" into an inverted L-shape, as shown in Figure 8b

Algorithm 4 is an recursive algorithm that takes three inputs: *currPath*, the path traversed so far; *Paths*, a collection of all generated paths; and n, the grid size. It checks the position of last point

Algorithm 1 Computing Zigzag-Partition of a Path

in: A path *zigzagPath* is an array of coordinates, where $v_i(x_i, y_i)$ is an *ith* coordinate
out: An array *partitionPoints* containing points of partitioning

function ZIGZAG-PARTITION(*zigzagPath*)
    Initialize *partitionPoints* ← []
    for $i \leftarrow 2$ to size(*zigzagPath*) − 1 **do**
        if Point $v_i(x_i, y_i)$ lies on the line $y = x + 1$ **then**
            if $v_{i-1} \rightarrow v_i$ is an Up step **then**
                Append i to *partitionPoints*
            if $v_i \rightarrow v_{i+1}$ is an *Right* step **then**
                Append i to *partitionPoints*
    return *partitionPoints*

Algorithm 2 Transform Partition of a Zigzag Path

in: A path *zigzagPath* is an array of coordinates, where $v_i(x_i, y_i)$ is an *ith* coordinate.
in: l and r are indices of the start and end points of the portion to be transformed.
out: The transformed *zigzagPath*.

function TRANSFORM-PARTITION(*zigzagPath*, l, r)
    for $i \leftarrow l + 1$ to $\lfloor (l + r)/2 \rfloor - 1$ **do**
        $zigzagPath[i] \leftarrow zigzagPath[i - 1] + (0, 1)$
    for $i \leftarrow \lfloor (l + r)/2 \rfloor$ to $r - 1$ **do**
        $zigzagPath[i] \leftarrow zigzagPath[i - 1] + (1, 0)$

Algorithm 3 Apply Transformation to a $x + 2$ Constrained Path

in: A path *constrainedPath* is an array of coordinates.
out: The transformed *constrainedPath* pattern.

function APPLY-TRANSFORMATION(*constrainedPath*)
    partitionPoints ← ZIGZAG-PARTITION(*constrainedPath*)
    for $i \leftarrow 0$ to $|partitionPoints| - 1$, **step** $i = i + 2$ **do**
        TRANSFORM-PARTITION(*constrainedPath*, *partitionPoints*[i], *partitionPoints*[$i + 1$])

Algorithm 4 Generate $x + 2$ Constrained Paths for an $n \times n$ grid

in: *CurrPath* refers to the path that has been traversed until now, *Patterns* contains $x + 2$ constrained paths from $(0, 0)$ to (n, n), n is the dimension of our square grid.

function GENERATEPATHS(*currPath*, *Paths*, n)
    $length \leftarrow |currPath|$
    $[a, b] \leftarrow currPath[length]$
    if $a > n$ or $b > n$ **then**
        return
    if $a = n$ and $b = n$ **then**
        Append *currPath* to *Paths*
    if $(a + 1, b)$ lies on or below the line $y = x + 2$ **then**
        NewPath ← **copy***currPath*
        Append $(a + 1, b)$ to *NewPath*
        GENERATEPATHS(*NewPath*,*Paths*,*n*)
    if $(a, b + 1)$ lies on or below the line $y = x + 2$ **then**
        NewPath ← **copy***currPath*
        Append $(a, b + 1)$ to *NewPath*
        GENERATEPATHS(*NewPath*,*Paths*,*n*)

in currPath and terminates the branch if it is outside the grid, if the point reaches the destination, *currPath* is appended to *Paths*, and then the branch terminates. Otherwise, the algorithm branches to two potential moves: a right step or an upward step , provided the new point lies on or below the line $y = x + 2$. For each valid move, a copy of *currPath* is created, the new point is appended, and the function is called recursively with the updated path. This process ensures that all valid $x + 2$ *constrained paths* are generated and stored in *Paths*. To convert these paths to *Right-Then-Up constrained paths*, we call the Algorithm 3 for each path in *Paths*. The total number of $x + 2$ *constrained path*, $|\mathbb{P}|$, is given by

$$|\mathbb{P}| = \binom{2n}{n} - \binom{2n}{n-3}.$$

Using asymptotic approximations,

$$|\mathbb{P}| \sim \frac{9 \cdot 4^n}{\sqrt{\pi} n^{3/2}}.$$

The cost of generating a single path is $\mathcal{O}(n)$. Thus, the overall time complexity for generating all paths is

$$\mathcal{O}(|\mathbb{P}| \cdot n) = \mathcal{O}\left(\frac{4^n}{n^{3/2}} \cdot n\right) = \mathcal{O}\left(\frac{4^n}{n^{1/2}}\right).$$

Since the transformation of each path (Algorithm 3) also requires $\mathcal{O}(n)$ time, the overall complexity for generating *Right-Then-Up constrained paths* remains

$$\mathcal{O}\left(\frac{4^n}{n^{1/2}}\right).$$

6. Conclusion

In this paper, we introduced an algorithm to generate a class of special lattice paths within a grid $n \times n$, which is particularly useful to identify and analyze short cycles in undirected bipartite graphs. A central contribution of this work is to establish a bijection between $x + 2$ *constrained paths* and *Right-Then-Up constrained paths*. We demonstrated that the generation and transformation processes for these paths can be accomplished with an overall computational complexity of $\mathcal{O}\left(\frac{4^n}{n^{1/2}}\right)$. These paths are equivalent to patterns that are used to find cycles in bipartite graphs [3]. This work can be used to extend the algorithm to find short cycles over a wider range of lengths. This study establishes a solid foundation for further exploration of lattice path transformations and their utility in solving complex combinatorial problems.

References

[1] R. Gallager, Low-density parity-check codes, IRE Transactions on Information Theory 8 (1962) 21–28. doi:10.1109/TIT.1962.1057683.

[2] R. Tanner, A recursive approach to low complexity codes, IEEE Transactions on Information Theory 27 (1981) 533–547. doi:10.1109/TIT.1981.1056404.

[3] A. Dehghan, A. H. Banihashemi, Counting short cycles in bipartite graphs: A fast technique/algorithm and a hardness result, IEEE Transactions on Communications 68 (2020) 1378–1390. URL: https://doi.org/10.1109/TCOMM.2019.2962397.

[4] M. Karimi, A. H. Banihashemi, Message-passing algorithms for counting short cycles in a graph, IEEE Transactions on Communications 61 (2013) 485–495. doi:10.1109/TCOMM.2012.100912.120503.

[5] J. Li, S. Lin, K. Abdel-Ghaffar, Improved message-passing algorithm for counting short cycles in bipartite graphs, in: 2015 IEEE International Symposium on Information Theory (ISIT), 2015, pp. 416–420. doi:10.1109/ISIT.2015.7282488.

[6] I. F. Blake, S. Lin, On short cycle enumeration in biregular bipartite graphs, IEEE Transactions on Information Theory 64 (2018) 6526–6535. doi:10.1109/TIT.2017.2784839.

[7] R. P. Stanley, S. Fomin, Enumerative Combinatorics, Cambridge Studies in Advanced Mathematics, Cambridge University Press, 1999. URL: https://doi.org/10.1017/CBO9780511609589.

[8] C. Krattenthaler, Lattice path enumeration, 2017. URL: https://arxiv.org/abs/1503.05930. arXiv:1503.05930.