Exact Enumeration of Multiple Self-Avoiding Polygons

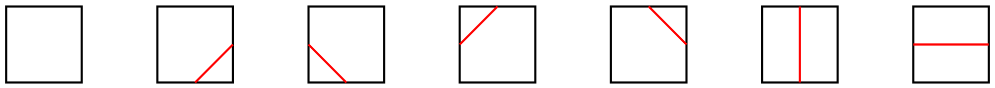
Jack Hanke, Richard Schank

Abstract

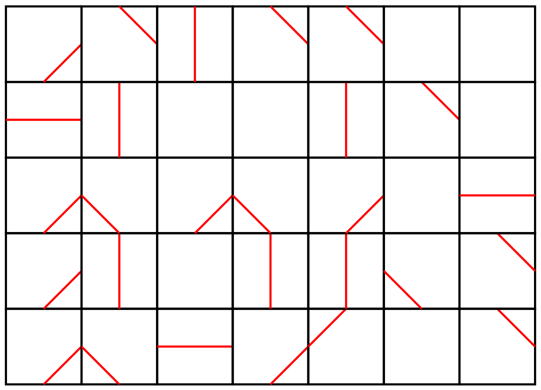
Hong and Oh calculated upper and lower bounds on the number of multiple self-avoiding polygons in the square lattice. These results accompanied 7 distinct tiles that together model multiple ring polymers in physics. We exactly enumerate these multiple self-avoiding polygons, along with a variation on the tile set and rules considered.

1 Introduction

Consider the following 7 unit squares with red lines on them.

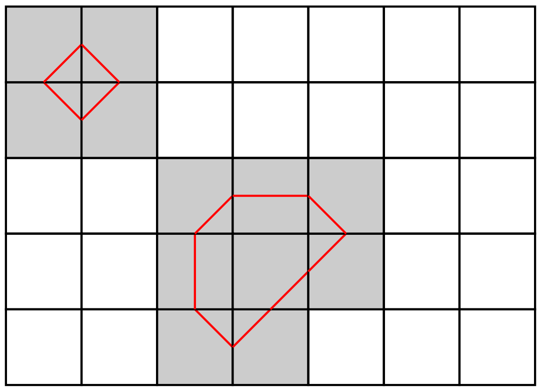


Call these unit squares *mosaic tiles*. Then let an (n, m)-mosaic be an $n \times m$ matrix of mosaic tiles. For example, below is a $(7, 5)$-mosaic.



Next, define a *polygon mosaic* as a mosaic in which any pair of mosaic tiles lying immediately next to each other in either the same row or the same column have or do not have connection points simultaneously on their common edge, as well as no connection point on the boundary edges.

Example 1.1. Below is an example of a polygon $(7, 5)$-mosaic that contains 2 polygons, hilighted in gray.



Hong and Oh [1, Hong2018] gave the following upper and lower bounds on the number of polygon (n,m)-mosaics $p_{n,m}+1$ (A181245 [2, OEIS]), namely

$$2^{n+m-3}\left(\frac{17}{10}\right)^{(n-2)(m-2)} \leq p_{n,m}+1 \leq 2^{n+m-3}\left(\frac{31}{16}\right)^{(n-2)(m-2)}.$$

We give an exact expression for $p_{n,m}+1$ in Theorem 1. We also introduce and enumerate *messy polygon mosaics*, a variant on the previously studied polygon mosaics with a similar enumeration strategy.

Theorem 1. *Let* $M(2)=\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}$ *for* $n\geq 2$. *Then for* $m \geq 2$, *if* $M(m)=\begin{bmatrix}M_1 & M_2\\ M_3 & M_4\end{bmatrix}$, *define*

$$M(m+1)=\begin{bmatrix}M_1 & M_2 & M_1 & M_2\\ M_3 & M_4 & 0M_3 & M_4\\ M_1 & 0M_2 & M_1 & M_2\\ M_3 & M_4 & M_3 & M_4\end{bmatrix},$$

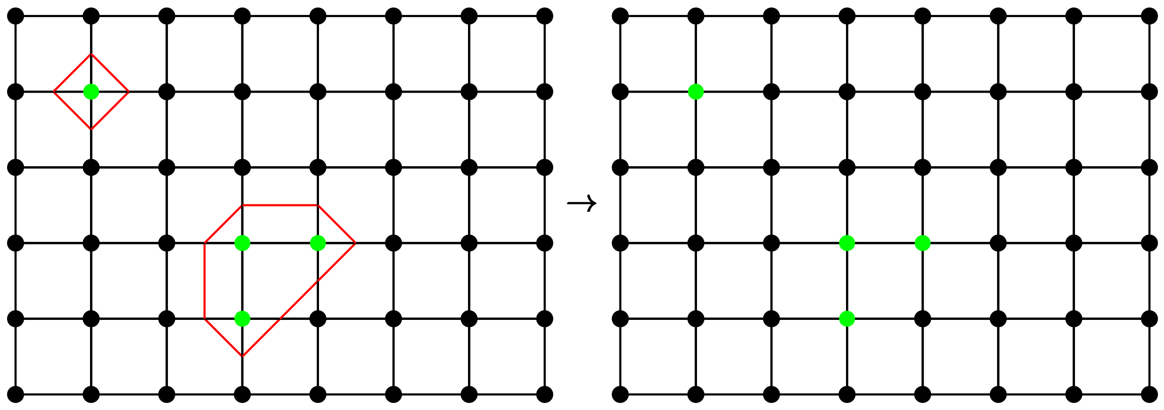
where M_i *is a sub-matrix of the block matrix* $M(m)$. *Next define the rows and columns of* $M(m)$ *as*

$$M(m)=\begin{bmatrix}r_1\\ r_2\\ \dots\\ r_{2^{m-1}}\end{bmatrix}=\begin{bmatrix}c_1 & c_2 & \dots & c_{2^{m-1}}\end{bmatrix}$$

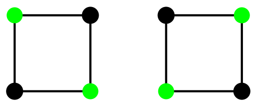
Then $p_{n,m}+1=r_1M(m)^{n-2}c_1$.

2 Proof of Theorem 1

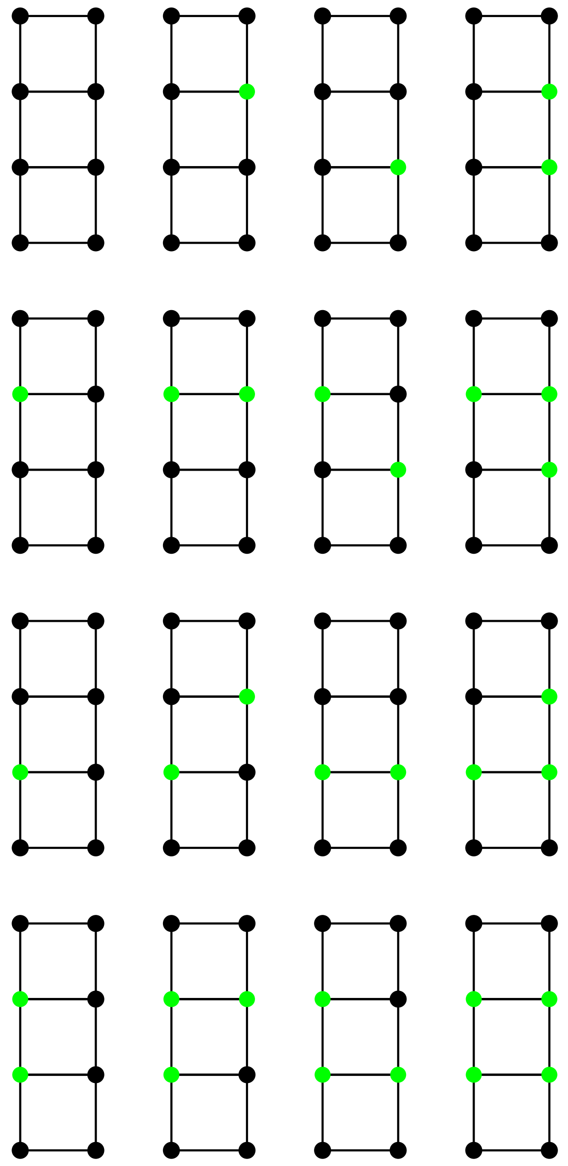
Proof. Begin by labelling the vertices of the rectangular lattice as follows. If the vertex is surrounded by an even number of SAPs, color it black. If the vertex is surrounded by an odd number of SAPs, color it green. Using the mosaic from Example 1.1, we get the following labelling.



Notice that vertices on the boundary of the lattice will always be black. The associated *parity configuration* for the above mosaic is below. The number of parity configurations that do not include the following sub-configurations also count the number of self-avoiding polygons.



We seek to construct a transition matrix for a fixed m that maps all possible parity configuration edge boundaries to every other. As an example, consider $m = 3$.



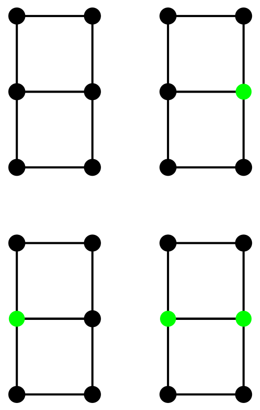
$$\rightarrow \qquad \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

We design $M(3)$ so that each column has the same right edge parity and every row has the same left edge parity. The entries of $M(3)$ are 1 if the parity configuration admits a tile configuration, and 0 otherwise.

If we define $M(3)$ in this way, then c_1 of $M(3)$ represents all parity configurations that "start with a boundary" of a parity configuration, and r_1 of $M(3)$ represents all parity configurations that "end with a boundary" of a parity configuration, if you build the parity configuration from right to left. This immediately gives that $p_{n,3} + 1 = r_1 M(3)^{n-2} c_1$.

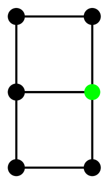
If $M(m)$ can be constructed for a fixed m, then the same argument can be used to enumerate $p_{n,m} + 1$. But how does one construct $M(m)$ for a general m?

Consider $M(2)$ along with its represented parity configurations below.

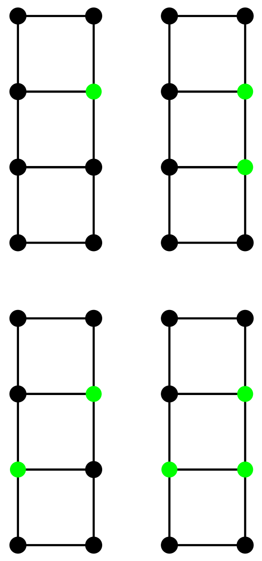


$$\rightarrow \quad \begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}$$

If one knows the $M(2)$ weight at, say, position $(0,1)$, then one can determine the mapping by considering appending all possible cell parity configurations to the bottom of the column, like so.



$$\rightarrow$$



$$\rightarrow \quad \begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}$$

If we write $M(2) = \begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix} = \begin{bmatrix}M_1 & M_2\\ M_3 & M_4\end{bmatrix}$, and compute these mappings for all possible parity configurations, we get

$$M(3) = \begin{bmatrix}1M_1 & 1M_2 & 1M_1 & 1M_2\\ 1M_3 & 1M_4 & 0M_3 & 1M_4\\ 1M_1 & 0M_2 & 1M_1 & 1M_2\\ 1M_3 & 1M_4 & 1M_3 & 1M_4\end{bmatrix}$$

.

Finally, as appending any parity configuration to the bottom of a column only changes the identity of the bottom two parity cell configurations, we can write that for any matrix $M(m) = \begin{bmatrix}M_1 & M_2\\ M_3 & M_4\end{bmatrix}$ we have

$$M(m+1) = \begin{bmatrix} 1M_1 & 1M_2 & 1M_1 & 1M_2 \\ 1M_3 & 1M_4 & 0M_3 & 1M_4 \\ 1M_1 & 0M_2 & 1M_1 & 1M_2 \\ 1M_3 & 1M_4 & 1M_3 & 1M_4 \end{bmatrix},$$

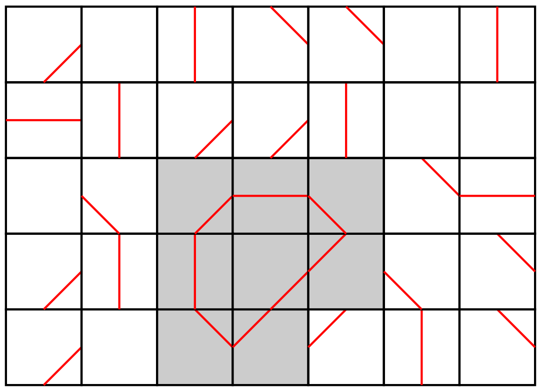
which completes the proof.

□

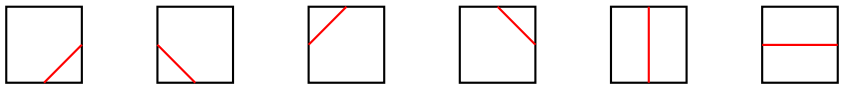
3 Messy Polygon Mosaics

Define a *messy polygon mosaic* as a mosaic that contains at least one self-avoiding polygon, with no restriction on other connection points.

Example 3.1. Below is a messy polygon $(7, 5)$-mosaic with 1 SAP hilighted in gray.

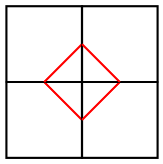


Note that one can also consider creating messy polygon mosaics with the alternate tile set below, where the empty tile is removed.



Due to the fact that the empty tile cannot contribute to the structure of the SAP, enumerating messy polygon mosaics under either tileset can be written with a parameter b, where $b = 6$ or $b = 7$ for the respective tileset. We work with the general b for the remainder of the paper.

Additionally, it turns out that it is easier to enumerate the number of messy mosaics that *do not* contain a SAP. Therefore, let $t_{n,m}$ be the number of mosaics that do not contain a SAP. Clearly $t_{n,m} = t_{m,n}$. Also from the fact that the smallest SAP is



we have that $t_{n,1} = b^n$, and $t_{2,2} = b^4 - 1$. It turns out that using similar techniques to Theorem 1 we can exactly enumerate messy mosaics.

Theorem 2. *Let $M(2) = \begin{bmatrix} b^2 & 1 \\ -1 & 1 \end{bmatrix}$ for $n \geq 2$, where $b = 6$ for the 6-tileset and $b = 7$ for the 7-tileset.Then for $m \geq 2$, if $M(m) = \begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix}$, define*

$$M(m+1) = \begin{bmatrix} bM_1 & bM_2 & \frac{1}{b}M_1 & M_2 \\ bM_3 & bM_4 & 0M_3 & M_4 \\ -\frac{1}{b}M_1 & 0M_2 & \frac{1}{b}M_1 & M_2 \\ M_3 & M_4 & -M_3 & bM_4 \end{bmatrix},$$

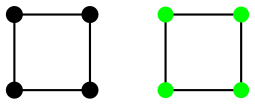
where M_i is a sub-matrix of the block matrix $M(m)$. For the given m, define the rows and columns of $M(m)$ as

$$M(m) = \begin{bmatrix} r_1 \\ r_2 \\ \dots \\ r_{2^{m-1}} \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & \dots & c_{2^{m-1}} \end{bmatrix}$$

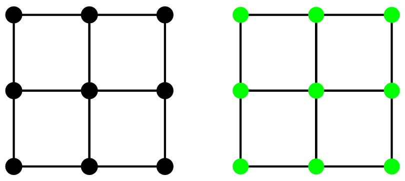
Then $t_{n,m} = r_1 M(m)^{n-2} c_1$.

4 Proof of Theorem 2

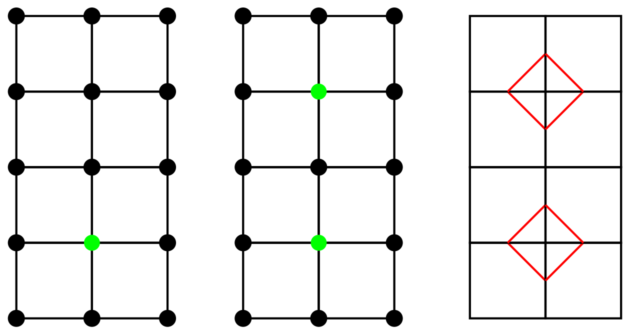
Proof. We again consider the parity configurations for a specific (n, m)-mosaic. However, there is not a bijection between messy polygon mosaics and parity configurations in the same way as with traditional polygon mosaics. This is because the following cell-parity configurations are no longer uniquely determined.



Additionally, the sub parity configurations shown below

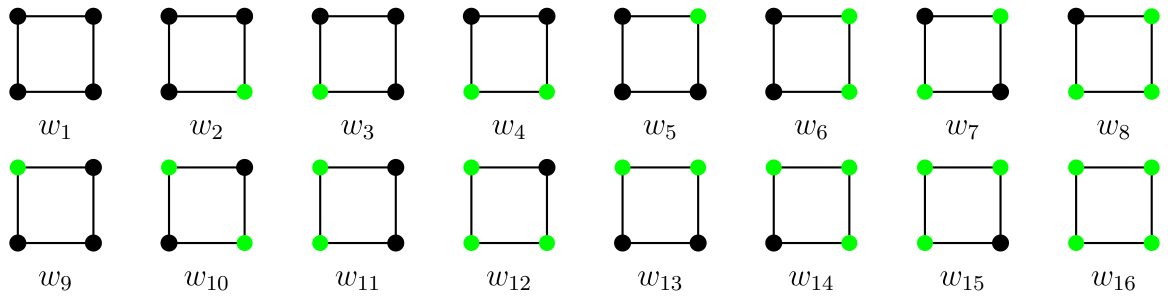


are now ambiguous as to whether or not they represent a SAP. Critically, both of the following parity configurations could include the right-most messy polygon mosaic.



In fact, the number of SAPs that appear in a given messy polygon mosaic is exactly the number of parity configurations that map to that mosaic. Therefore, if we are to count these mosaics, we need to account for this double-counting with the inclusion-exclusion principle.

To create an analagous result to Theorem 1 we need to find a weight assignment to the 16 parity configurations below that fulfill a set of conditions.



First $w_7 = w_{10} = 0$, as again these are impossible parity configurations for our tile set. Next notice that $w_1 = w_{16} = b$, as these cells do not indicate a specific tile. The remaining weights uniquely specify a tile, and so are equal to 1 or -1. But how do we find these assignments?

First notice that we want a weight assignment so that the parity configurations for a given SAP multiply to -1. This is due to the inclusion-exclusion principle. This means that if there are multiple SAPs in a mosaic, then the product will be positive if there is an even number of SAPs specified, and negative if there is an odd number specified.

Next note the following lemma.

Lemma 3. *One can construct all larger SAPs from the smallest SAP using a finite set of transformations S.*

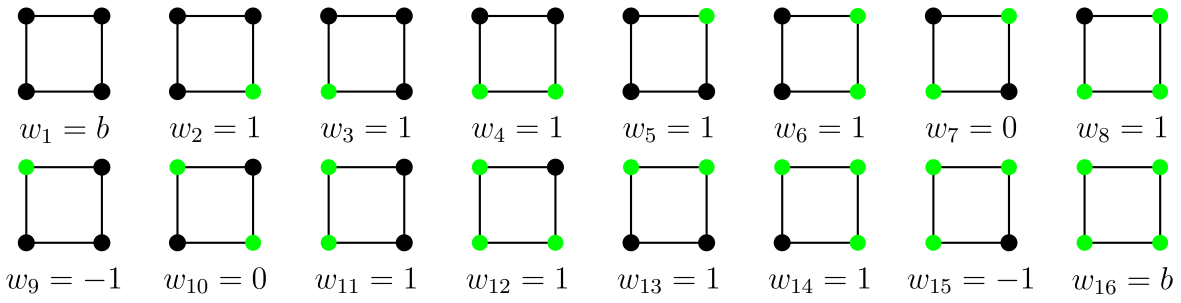
Proof. TODO □

This is because one can find $w_1, \dots, w_{16}$ so that the following two constraints hold:

Constraint 4. *The weights associated with the smallest SAP multiply to -1, ie. $w_2 w_3 w_5 w_9 = -1$.*

Constraint 5. All *transformations in S preserve the weight product of a changed SAP.*

Constraint 4 and Constraint 5 amount to a series of constraints on the values of w_i. The derivation for these constraints can be found in the Appendix. Choosing a solution set from these constraints gives the following weights.



This immediately gives us a way to construct an analagous definition for our transition matrixes $M(m) = \begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix}$. As the appending of a cell parity configuration can possibly change the number of admitted cells, we define $v_1, \ldots, v_{16}$ for the associated change in the number of mosaics for the recursive definition of $M(m+1)$. A simple way to obtain $v_1, \ldots, v_{16}$ is to directly compute $M(2)$ and $M(3)$ and compare their values like so.

$$M(2) = \begin{bmatrix} b^2 & 1 \\ -1 & 1 \end{bmatrix}, M(3) = \begin{bmatrix} b^3 & b & b & 1 \\ -b & b & 0 & 1 \\ -b & 0 & b & 1 \\ -1 & 1 & 1 & b \end{bmatrix}.$$

This gives

$$M(m+1) = \begin{bmatrix} v_1 M_1 & v_2 M_2 & v_3 M_1 & v_4 M_2 \\ v_5 M_3 & v_6 M_4 & v_7 M_3 & v_8 M_4 \\ v_9 M_1 & v_{10} M_2 & v_{11} M_1 & v_{12} M_2 \\ v_{13} M_3 & v_{14} M_4 & v_{15} M_3 & v_{16} M_4 \end{bmatrix} = \begin{bmatrix} bM_1 & bM_2 & \frac{1}{b} M_1 & M_2 \\ bM_3 & bM_4 & 0M_3 & M_4 \\ -\frac{1}{b} M_1 & 0M_2 & \frac{1}{b} M_1 & M_2 \\ M_3 & M_4 & -M_3 & bM_4 \end{bmatrix}.$$

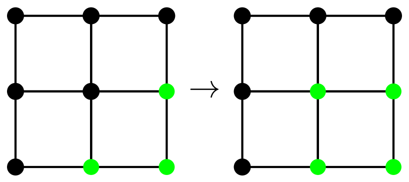
An analagous argument to Theorem 1 gives the result.

□

5 Appendix

Flipping the parity of a single vertex in a parity configuration changes the 4 surrounding cells. This creates a constraint on a subset of $w_1, \ldots, w_{16}$.

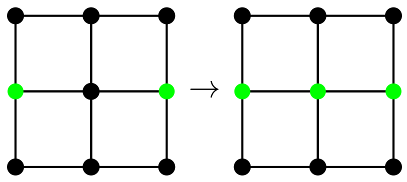
The flipping of parity of a single vertex can result in 2 distinct types of constraints. Let a constraint of *Type 1* be a parity flip that does not change the number of SAPs in the parity configuration. For example, consider the following flip of the center vertex in the following portion of a parity configuration.



As this does not change the associated number of SAPs in the larger parity configuration, we want this to preserve the sign of the weight product. This gives the following associated constraint.

$$\mathrm{sign}(w_1 w_2 w_5 w_9) = \mathrm{sign}(w_2 w_4 w_6 w_{16}).$$

Now let a constraint of *Type 2* be a parity flip that does change the number of SAPs. For example, consider flipping the center vertex of the following portion of a parity configuration.



The above transformation corresponds with *either* two distinct SAPs joining into one SAP *or* one SAP splitting into two distinct SAPs. In either case, we want the sign of the product to switch. This corresponds with the following constraint.

$$\mathrm{sign}(w_3 w_2 w_9 w_5) = -\mathrm{sign}(w_4 w_4 w_{13} w_{13}).$$

All Type 1 constraints are as follows. For the following set of equations, assume the equals sign (=) means *only* equal in sign.

$$\begin{array}{rrr}
w_1w_1w_2w_3 = w_2w_3w_6w_{11} & w_1w_1w_2w_4 = w_2w_3w_6w_{12} & w_1w_1w_4w_3 = w_2w_3w_8w_{11} \\
w_1w_1w_4w_4 = w_2w_3w_8w_{12} & w_1w_2w_1w_5 = w_2w_4w_5w_{13} & w_1w_2w_1w_6 = w_2w_4w_5w_{14} \\
w_1w_2w_2w_8 = w_2w_4w_6w_{16} & w_1w_2w_4w_8 = w_2w_4w_8w_{16} & w_3w_1w_9w_1 = w_4w_3w_{13}w_9 \\
w_3w_1w_{11}w_1 = w_4w_3w_{15}w_9 & w_3w_1w_{12}w_3 = w_4w_3w_{16}w_{11} & w_3w_1w_{12}w_4 = w_4w_3w_{16}w_{12} \\
w_3w_2w_{12}w_8 = w_4w_4w_{16}w_{16} & w_1w_6w_1w_5 = w_2w_8w_5w_{13} & w_1w_6w_1w_6 = w_2w_8w_5w_{14} \\
w_1w_6w_2w_8 = w_2w_8w_6w_{16} & w_1w_6w_4w_8 = w_2w_8w_8w_{16} & w_3w_6w_{12}w_8 = w_4w_8w_{16}w_{16} \\
w_5w_9w_1w_1 = w_6w_{11}w_5w_9 & w_5w_{13}w_1w_1 = w_6w_{15}w_5w_9 & w_5w_{14}w_1w_5 = w_6w_{16}w_5w_{13} \\
w_5w_{14}w_1w_6 = w_6w_{16}w_5w_{14} & w_5w_{14}w_2w_8 = w_6w_{16}w_6w_{16} & w_5w_{14}w_4w_8 = w_6w_{16}w_8w_{16} \\
w_{11}w_1w_9w_1 = w_{12}w_3w_{13}w_9 & w_{11}w_1w_{11}w_1 = w_{12}w_3w_{15}w_9 & w_{11}w_1w_{12}w_3 = w_{12}w_3w_{16}w_{11} \\
w_{11}w_1w_{12}w_4 = w_{12}w_3w_{16}w_{12} & w_{11}w_2w_{12}w_8 = w_{12}w_4w_{16}w_{16} & w_{11}w_6w_{12}w_8 = w_{12}w_8w_{16}w_{16} \\
w_{13}w_9w_1w_1 = w_{14}w_{11}w_5w_9 & w_{15}w_9w_9w_1 = w_{16}w_{11}w_{13}w_9 & w_{15}w_9w_{11}w_1 = w_{16}w_{11}w_{15}w_9 \\
w_{15}w_9w_{12}w_3 = w_{16}w_{11}w_{16}w_{11} & w_{15}w_9w_{12}w_4 = w_{16}w_{11}w_{16}w_{12} & w_{13}w_{13}w_1w_1 = w_{14}w_{15}w_5w_9 \\
w_{13}w_{14}w_1w_5 = w_{14}w_{16}w_5w_{13} & w_{13}w_{14}w_1w_6 = w_{14}w_{16}w_5w_{14} & w_{13}w_{14}w_2w_8 = w_{14}w_{16}w_6w_{16} \\
w_{13}w_{14}w_4w_8 = w_{14}w_{16}w_8w_{16} & w_{15}w_{13}w_9w_1 = w_{16}w_{15}w_{13}w_9 & w_{15}w_{13}w_{11}w_1 = w_{16}w_{15}w_{15}w_9 \\
w_{15}w_{13}w_{12}w_3 = w_{16}w_{15}w_{16}w_{11} & w_{15}w_{13}w_{12}w_4 = w_{16}w_{15}w_{16}w_{12} & w_{15}w_{14}w_9w_5 = w_{16}w_{16}w_{13}w_{13} \\
w_{15}w_{14}w_9w_6 = w_{16}w_{16}w_{13}w_{14} & w_{15}w_{14}w_{11}w_5 = w_{16}w_{16}w_{15}w_{13} & w_{15}w_{14}w_{11}w_6 = w_{16}w_{16}w_{15}w_{14}
\end{array}$$

Similarly, all Type 2 constraints are as follows. Again, for the following set of equations, assume the equals sign (=) means *only* equal in sign.

$$\begin{aligned}
-w_3w_2w_9w_5 &= w_4w_4w_{13}w_{13} & -w_3w_2w_9w_6 &= w_4w_4w_{13}w_{14} & -w_3w_2w_{11}w_5 &= w_4w_4w_{15}w_{13}\\
-w_3w_2w_{11}w_6 &= w_4w_4w_{15}w_{14} & -w_3w_6w_9w_5 &= w_4w_8w_{13}w_{13} & -w_3w_6w_9w_6 &= w_4w_8w_{13}w_{14}\\
-w_3w_6w_{11}w_5 &= w_4w_8w_{15}w_{13} & -w_3w_6w_{11}w_6 &= w_4w_8w_{15}w_{14} & -w_5w_9w_2w_3 &= w_6w_{11}w_6w_{11}\\
-w_5w_9w_2w_4 &= w_6w_{11}w_6w_{12} & -w_5w_9w_4w_3 &= w_6w_{11}w_8w_{11} & -w_5w_9w_4w_4 &= w_6w_{11}w_8w_{12}\\
-w_5w_{13}w_2w_3 &= w_6w_{15}w_6w_{11} & -w_5w_{13}w_2w_4 &= w_6w_{15}w_6w_{12} & -w_5w_{13}w_4w_3 &= w_6w_{15}w_8w_{11}\\
-w_5w_{13}w_4w_4 &= w_6w_{15}w_8w_{12} & -w_{11}w_2w_9w_5 &= w_{12}w_4w_{13}w_{13} & -w_{11}w_2w_9w_6 &= w_{12}w_4w_{13}w_{14}\\
-w_{11}w_2w_{11}w_5 &= w_{12}w_4w_{15}w_{13} & -w_{11}w_2w_{11}w_6 &= w_{12}w_4w_{15}w_{14} & -w_{11}w_6w_9w_5 &= w_{12}w_8w_{13}w_{13}\\
-w_{11}w_6w_9w_6 &= w_{12}w_8w_{13}w_{14} & -w_{11}w_6w_{11}w_5 &= w_{12}w_8w_{15}w_{13} & -w_{11}w_6w_{11}w_6 &= w_{12}w_8w_{15}w_{14}\\
-w_{13}w_9w_2w_3 &= w_{14}w_{11}w_6w_{11} & -w_{13}w_9w_2w_4 &= w_{14}w_{11}w_6w_{12} & -w_{13}w_9w_4w_3 &= w_{14}w_{11}w_8w_{11}\\
-w_{13}w_9w_4w_4 &= w_{14}w_{11}w_8w_{12} & -w_{13}w_{13}w_2w_3 &= w_{14}w_{15}w_6w_{11} & -w_{13}w_{13}w_2w_4 &= w_{14}w_{15}w_6w_{12}\\
-w_{13}w_{13}w_4w_3 &= w_{14}w_{15}w_8w_{11} & -w_{13}w_{13}w_4w_4 &= w_{14}w_{15}w_8w_{12} & &
\end{aligned}$$

Solving all Type 1 and Type 2 constraints gives the following solution set.

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}	w_{12}	w_{13}	w_{14}	w_{15}	w_{16}
b	-1	-1	-1	-1	-1	0	-1	1	0	-1	-1	-1	-1	1	b
b	-1	-1	-1	-1	1	0	1	1	0	1	1	-1	1	-1	b
b	-1	-1	-1	1	-1	0	-1	-1	0	-1	-1	-1	1	-1	b
b	-1	-1	-1	1	1	0	1	-1	0	1	1	-1	-1	1	b
b	-1	-1	1	-1	-1	0	1	1	0	-1	1	1	1	-1	b
b	-1	-1	1	-1	1	0	-1	1	0	1	-1	1	-1	1	b
b	-1	-1	1	1	-1	0	1	-1	0	-1	1	1	-1	1	b
b	-1	-1	1	1	1	0	-1	-1	0	1	-1	1	1	-1	b
b	-1	1	-1	-1	-1	0	-1	-1	0	-1	1	-1	-1	-1	b
b	-1	1	-1	-1	1	0	1	-1	0	1	-1	-1	1	1	b
b	-1	1	-1	1	-1	0	-1	1	0	-1	1	-1	1	1	b
b	-1	1	-1	1	1	0	1	1	0	1	-1	-1	-1	-1	b
b	-1	1	1	-1	-1	0	1	-1	0	-1	-1	1	1	1	b
b	-1	1	1	-1	1	0	-1	-1	0	1	1	1	-1	-1	b
b	-1	1	1	1	-1	0	1	1	0	-1	-1	1	-1	-1	b
b	-1	1	1	1	1	0	-1	1	0	1	1	1	1	1	b
b	1	-1	-1	-1	-1	0	1	-1	0	-1	-1	-1	-1	-1	b
b	1	-1	-1	-1	1	0	-1	-1	0	1	1	-1	1	1	b
b	1	-1	-1	1	-1	0	1	1	0	-1	-1	-1	1	1	b
b	1	-1	-1	1	1	0	-1	1	0	1	1	-1	-1	-1	b
b	1	-1	1	-1	-1	0	-1	-1	0	-1	1	1	1	1	b
b	1	-1	1	-1	1	0	1	-1	0	1	-1	1	-1	-1	b
b	1	-1	1	1	-1	0	-1	1	0	-1	1	1	-1	-1	b
b	1	-1	1	1	1	0	1	1	0	1	-1	1	1	1	b
b	1	1	-1	-1	-1	0	1	1	0	-1	1	-1	-1	1	b
b	1	1	-1	-1	1	0	-1	1	0	1	-1	-1	1	-1	b
b	1	1	-1	1	-1	0	1	-1	0	-1	1	-1	1	-1	b
b	1	1	-1	1	1	0	-1	-1	0	1	-1	-1	-1	1	b
b	1	1	1	-1	-1	0	-1	1	0	-1	-1	1	1	-1	b
b	1	1	1	-1	1	0	1	1	0	1	1	1	-1	1	b
b	1	1	1	1	-1	0	-1	-1	0	-1	-1	1	-1	1	b
b	1	1	1	1	1	0	1	-1	0	1	1	1	1	-1	b

Any of these assignments are sufficient for calculating $t_{n,m}$.

References

[1] Kyungpyo Hong and Seungsang Oh. “Bounds on Multiple Self-avoiding Polygons”. In: *Canadian Mathematical Bulletin* 61.3 (Sept. 2018), pp. 518–530. ISSN: 1496-4287. DOI: 10.4153/cmb-2017-072-x. URL: http://dx.doi.org/10.4153/CMB-2017-072-x.

[2] OEIS Foundation Inc. *The On-Line Encyclopedia of Integer Sequences*. Published electronically at http://oeis.org.