Knot Probabilities in Random Diagrams

Jason Cantarella, Harrison Chapman* and Matt Mastin†

We consider a natural model of random knotting– choose a knot diagram at random from the finite set of diagrams with n crossings. We tabulate diagrams with 10 and fewer crossings and classify the diagrams by knot type, allowing us to compute exact probabilities for knots in this model. As expected, most diagrams with 10 and fewer crossings are unknots (about 78% of the roughly 1.6 billion 10 crossing diagrams). For these crossing numbers, the unknot fraction is mostly explained by the prevalence of "tree-like" diagrams which are unknots for any assignment of over/under information at crossings. The data shows a roughly linear relationship between the log of knot type probability and the log of the frequency rank of the knot type, analogous to Zipf's law for word frequency. The complete tabulation and all knot frequencies are included as supplementary data.

1.   INTRODUCTION

The study of random knots goes back to the 1960's, when polymer physicists realized that the knot type of a closed (or ring) polymer would play an important role in the statistical mechanics of the polymer [14]. Orlandini and Whittington [30] give a comprehensive survey of the development of the field in the subsequent years. Most random knot models are based on closed random walks, and there are many variants corresponding to different types of walks (lattice walks, random walks with fixed edgelengths, random walks with variable edgelengths, random walks with different types of geometric constraints such as fixed turning angles). For almost all of these models, certain phenomena have been rigorously established– for instance, the probability of knotting goes to 1 exponentially fast as the size of the walk increases[12, 31, 35]. However, it has been difficult to prove more informative theorems.

One way to look at the problem is these knot models are all models of random space curves, and it is quite hard to relate the three-dimensional shape of a space curve to its knot type. One can in principle express the finite-type invariants as (complicated) integrals in the spirit of [24], but so far computing the expected value of these integrals has been too difficult.

The point of this paper is to look at knot diagrams not as convenient combinatorial representations of space curves, but as a probability space in their own right. This is basically the same model of random knotting as [13] or [11]: the objects are equivalence classes of immersions of S^1 into S^2 (as in Arnol'd [3]) paired with assignments of orientation and crossing signs. There are other quite

---

*University of Georgia, Mathematics Department, Athens GA
†MailChimp, Atlanta, GA

different combinatorial models of random knotting in the literature– see [8], [15], or [29].

Previous authors [11] have sampled a slightly different version of this space by using an algorithm of Schaeffer and Zinn-Justin [34] to uniformly sample a space of marked link diagrams and keeping only the knots. We wanted to get exact values for probabilities and also to study the probability of very rare knots, so a sampling approach was not appropriate for our purposes. Instead, we carried out a complete enumeration of knot diagrams up to 10 crossings using the graph theory software *plantri* [5, 6].

This paper is primarily computational. While we give a proof that our algorithm for enumerating diagrams is correct, our main contribution is the data set of diagrams itself. To generate the diagrams, we first enumerated the unoriented immersions of S^1 into S^2 in two different ways, as described below. We call these immersions knot shadows[1]. The sets of shadows generated in each way were identical, and we checked the number of shadows against counts by Arnol'd for $n \leq 5$ [3, page 79], sequence A008989 in the Online Encyclopedia of Integer Sequences [1], and later enumerations of Kapolnái et al. [21] and Coqueraux et al. [9]. We then enumerate assignments of crossing information and orientation for each shadow up to diffeomorphisms of S^2 preserving the orientation of the curve. (This is more complicated than it first seems; if the diagram has a symmetry, not all assignments of crossing information are different.) Finally, we compute the knot type for each assignment of crossing and orientation. These computations are new and fairly substantial. Enumerating the 1.6 billion 10-crossing knot diagrams and computing their HOMFLY-PT polynomial took several thousand hours of CPU time on the Amazon EC2 cloud computing service.

Figure 16 shows the relative frequency of all the knot types we observed, sorted by rank order among knots. When plotted on a log-log plot, we see that the plot is roughly linear across 9 orders of magnitude– giving some evidence for a roughly Zipfian distribution of knot types. In such a distribution, the k-th most frequent knot type would have a probability proportional to k^{-s} for some s. It would be very interesting to know what happens for large n.

How many of these diagrams represent nontrivial knots? Relatively few. Even for 10-crossing diagrams, the proportion of unknots is about 77%. For these crossing numbers, this is largely explained by the surprising frequency of "tree-like" knot shadows, studied by Aicardi [2], (Figure 18) for which the knot type does not depend on the assignment of crossing information– the resulting knot is *always* the unknot. These diagrams are surprisingly common: 42.05% of 8-crossing diagrams are treelike, which explains about half of the unknots among 8-crossing diagrams (84% of 8-crossing diagrams are unknots). The remaining unknots are almost all connect sums of treelike diagrams with the unique prime 3- or 4-crossing shadow. We will show that a simple analysis (Proposition 19) based on such diagrams gives a lower bound of 77% for the unknot fraction in 8-crossing diagrams, explaining more than 90% of all unknots in this class.

Tree-like diagrams are composite diagrams where each prime summand is one-crossing knot

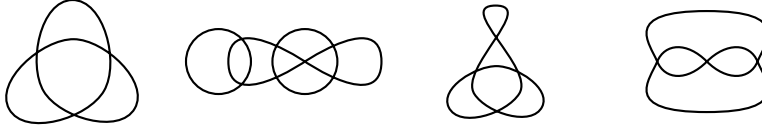---

[1] Kauffman [22] calls these knot *universes*.

FIG. 1: Examples of link shadows.

diagram ∞. In fact, random diagrams are in general highly composite [7] (Figure 18) with many simple summands. It remains an interesting open question to try to characterize the asymptotic distribution of sizes and numbers of connect summands in a random diagram for large n.

In general, counting knot diagrams is made significantly more complicated by the existence of diagram symmetries; for instance, a correction factor due to symmetries is the main technical difference between this natural model and the model of [34]. Our data shows that this difficulty rapidly disappears as the number of crossings increases; the average size of the automorphism group for a random 10 crossing knot diagram, for instance, is $846929/823832 \simeq 1.028$. This means that we expect this model to behave very much like the "rooted" model of [34] for crossing numbers 11 and above; one of us (Chapman) has shown this is true for sufficiently many crossings [7].

2. DEFINITIONS

We begin with some definitions.

Definition 1. We define a *link shadow L* with n vertices to be an equivalence class of generic smooth immersions of a collection of oriented circles into S^2 with n intersections up to diffeomorphism of the sphere (which may not be orientation-preserving on either the sphere or the circles; Arnol'd [3] calls this equivalence "OU").

Examples of link shadows are shown in Figure 1. In our experiments, each link shadow will be represented (non-uniquely) by a combinatorial object called a PD-code.

Definition 2. A **pd-code** $\mathfrak{L}$ with n-vertices is a list of n cyclically ordered quadruples called vertices which partition the set of signed edges $\{\pm 1, \ldots, \pm 2n\}$ so that in each quadruple every pair of non-adjacent edges have opposite signs.

Two pd-codes are **pd-isomorphic** if there exists a bijection of vertices and edges which respects the partition of signed edges into vertices, including globally preserving or reversing the cyclic ordering of edges around vertices. A pd-isomorphism must take both signs of an edge $\pm i$ to both signs of another edge $\pm j$, but can take $+i$ to $-j$ and vice versa.

For example $\mathfrak{L}_0 = ((+4, -2, -5, +1), (+2, -6, -3, +5), (+6, -4, -1, +3))$ is a valid pd-code of three vertices and six edges, as each of $\pm 1, \ldots, \pm 6$ occurs once, and each pair of non-

adjacent labels in a single quadruple (such as $+4, -5$ and $-2, +1$ in the first quadruple) has opposite signs.

This definition is similar to the "combinatorial maps" of Walsh and Lehman [36]. Indeed, a pd-code $\mathfrak{L}$ describes an element σ of the permutation group on the $4n$ letters $\{\pm 1, \pm 2, \cdots, \pm 2n\}$ which is a product of n disjoint cycles of size 4, together with an implicit permutation $\tau = (+1 - 1)(+2 - 2) \cdots (+2n - 2n)$ and a "consistent" choice of orientations on the edges. This (and other) formulations are explored by Coquereaux, et al. [9].

Definition 3. Given a pd-code $\mathfrak{L}$ and a signed edge e, we can define the *successor edge* $s(e)$ to be minus the edge immediately preceding e in the cyclic ordering of the quadruple where e occurs. The faces of a pd-code $\mathfrak{L}$ are the orbits of the successor map.

In the pd-code $\mathfrak{L}_0$, for example, -2 occurs in the quadruple $(+4, -2, -5, +1)$, so $s(-2) = -(+4) = -4$. Similarly, $s(s(-2)) = s(-4) = -(+6) = -6$, and $s(s(s(-2))) = s(-6) = -(+2) = -2$. Thus $(-2, -4, -6)$ is a face of $\mathfrak{L}_0$.

Proposition 4 (Mastin, [25]). *The vertices, edges, and faces of $\mathfrak{L}$ form a cell-complex structure on a 2-dimensional surface $C(\mathfrak{L})$. Given two pd-codes $\mathfrak{L}$ and $\mathfrak{L}'$, the pd-codes are pd-isomorphic $\iff$ the cell-complexes $C(\mathfrak{L})$ and $C(\mathfrak{L}')$ are isomorphic (the isomorphism may reverse orientation).*

We can then define

Definition 5. The *genus* g of a pd-code $\mathfrak{L}$ is the genus of the associated cell complex $C(\mathfrak{L})$. It is given by $V - E + F = 2 - 2g$, where V, E, and F count the vertices, edges, and faces of $\mathfrak{L}$. If a pd-code has genus 0, it is *planar*.

It is another theorem of Mastin that

Proposition 6 (Mastin [25]). *There is a bijection between n-vertex link shadows and n-vertex planar pd-codes up to pd-isomorphism.*

The bijection itself is easy to construct: the vertices of the pd-code represent intersections of the circles, the edges represent the sections of circles between intersections (signs represent orientation along the circle), and the cyclic ordering of the vertices is the counterclockwise order of circle arcs at each intersection point. An example of a link shadow and the associated pd-code is shown in Figure 2. It is clear that there are finitely many n-vertex planar pd-codes, and so finitely many equivalence classes of pd-codes up to isomorphism. By the proposition, there are therefore finitely many n-vertex link shadows. This will be our initial probability space.

Given a link shadow $\mathfrak{L}$ we can define a link diagram by

Definition 7. A *link diagram* is a link shadow where each intersection is decorated with over-under information for the circles meeting at the intersection. We call these intersections *crossings*. The

$$\{[+4, -2, -5, +1], [+2, -6, -3, +5],$$
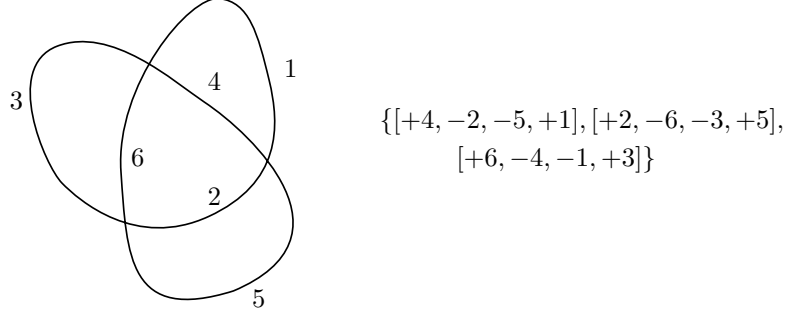$$[+6, -4, -1, +3]\}$$

FIG. 2: A diagram for 3_1 and its pd-code. There is only one component. Note that we may omit directional arrows as the orientation can be inferred from the ordering of the edge labels.

equivalence relation for diagrams is the *diagram isomorphism*: a diffeomorphism (not necessarily orientation-preserving) of S^2 to S^2 which respects over-under information and preserves the orientation of the curves.

It is clear that there are at most $2^{\# \text{ crossings}}$ link diagrams associated to a given link shadow, but that this number will be smaller if there are nontrivial shadow automorphisms.

Definition 8. In the *random diagram model*, a random n-crossing knot is selected uniformly from the counting measure on the finite set of one-component n-crossing link diagrams.

This is the smallest probability space including all the knot diagrams that one can define; however, note that we could expand the probability space by doing things like choosing planar, rather than spherical embeddings (hence labeling a face as the exterior) or choosing a basepoint on each component. These amount to making the rules for diagram and shadow isomorphism more strict, increasing the number of equivalence classes of diagrams.

3.   ENUMERATING SHADOWS

Our first goal is to enumerate the link shadows. This section describes our two enumeration algorithms. Each is built on the same computational foundation; a library which allows us to manipulate pd-codes. In this library, we provide

Definition 9. A *pdstor* is an ordered collection of pd-isomorphism types of pd-codes. We define the operation of adding a pd-code to a pdstor as adding the pd-code if it does not belong to an existing pd-isomorphism type (that is, is not pd-isomorphic to a pd-code present in the pdstor) and ignoring the pd-code otherwise.
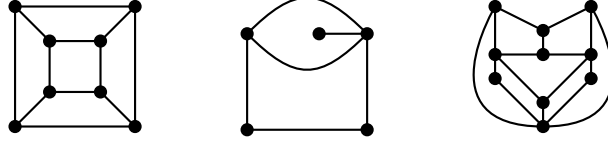
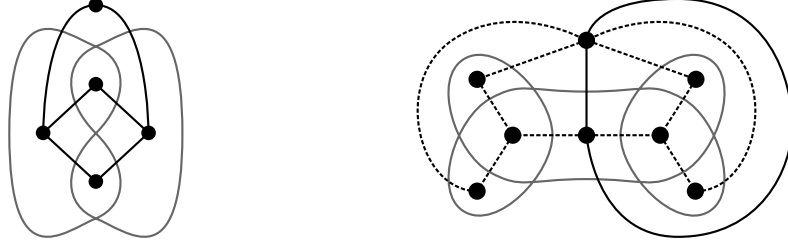FIG. 3: This figure shows examples of quadrangulations.

FIG. 4: This figure shows examples of quadrangulations and their dual graphs, which are link shadows. The quadrangulation at left is simple; the corresponding shadow is prime. The quadrangulation at right is non-simple; the corresponding shadow is composite.

Our implementation uses a combination of hashing and brute-force comparison to check whether a given pd-code is already pd-isomorphic to something in the database. Details of the pd-isomorphism check are provided in Appendix A for the curious reader.

We are left with the problem of creating a set of input pd-codes to the pdstor which are guaranteed to cover all n-vertex link shadows. We will then separate out the one-component knot shadows by searching the pdstor.

3.1. Dual quadrangulations and connect sums

Brinkmann et al. [5] provide an algorithm to enumerate all of the simple embedded planar quadrangulations of S^2, up to embedded isomorphism. A quadrangulation is a planar graph where each face has four edges, as in Figure 3. The dual graph to a quadrangulation is a connected 4-regular embedded planar multigraph (graphs embedded on a surface sometimes called "maps"), as shown at left in Figure 4. In other words, the dual graph defines a link shadow. This is almost enough to enumerate shadows. However, not every link shadow is obtained in this way: if the quadrangulation is simple, no pair of faces in the link shadow share more than one edge. As we can see in Figure 4, this property is not true for every link shadow. We need a familiar idea in a slightly new guise: prime and composite shadows.
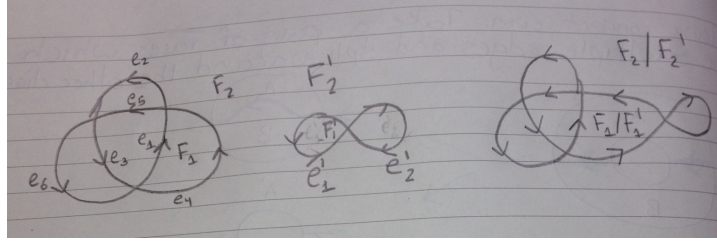
FIG. 5: The connected sum of two link shadows yields a new shadow where two pairs of faces have been merged.

Definition 10. Every pair of faces in a *prime shadow* shares at most one edge; all other link shadows are *composite shadows*.

This is not the same thing as prime and composite links– we can assign crossings and get a prime diagram of a composite link or a composite diagram of a prime link– but the theory is quite similar. The biggest difference, as we will see below, is that while connect sum for knot types is associative, connect sum for knot shadows is not.

Definition 11. Given edges e and e' in pd-codes $\mathfrak{L}$ and $\mathfrak{L}'$, we can construct a new pd-code $\mathfrak{L}\#_{e,e'}\mathfrak{L}'$. The edges of $\mathfrak{L}\#_{e,e'}\mathfrak{L}'$ are the edges of $\mathfrak{L}$ together with the edges of $\mathfrak{L}'$. The vertices of $\mathfrak{L}\#_{e,e'}\mathfrak{L}'$ are the vertices of $\mathfrak{L}$ together with the vertices of $\mathfrak{L}'$ with one change: $+e$ and $+e'$ are swapped ($-e$ and $-e'$ stay in their original positions).

The effect of the definition is to switch the positions of the heads of the edges e and e' in the crossings where they occur. We are not trying to be too specific about notation because the edges will need to be relabeled in the new pd-code (see our code for one possible implementation). We can then prove

Proposition 12. *If $\mathfrak{L}$ and $\mathfrak{L}'$ are planar, then so is the pd-code $\mathfrak{L}\#_{e,e'}\mathfrak{L}'$.*

Proof. The new collection of vertices clearly still partitions the new collection of signed edge labels, since the same edge labels occur in vertices as in the original pd-codes $\mathfrak{L}$ and $\mathfrak{L}'$. We are swapping a pair of $+$ labels, so the rule on non-adjacent signs is still obeyed in the new pd-code. This means that we need only check the genus.

If we think of faces of a pd-code as lists of signed edges, the effect of the connect sum operation is to concatenate two pairs of these lists. As Figure 5 shows, the successor of $+e$ in $\mathfrak{L}$ is replaced by the successor of $+e'$ in $\mathfrak{L}'$, and the chain of successors continues as before until returning eventually to the successor of $+e$ in $\mathfrak{L}$. This creates a new face in $\mathfrak{L}\#_{e,e'}\mathfrak{L}'$ which is formed by merging two previous faces in $\mathfrak{L}$ and $\mathfrak{L}'$ and contains $+e$ and $+e'$. We can argue similarly that a second new face of $\mathfrak{L}\#_{e,e'}\mathfrak{L}'$ is also created, containing $-e$ and $-e'$.
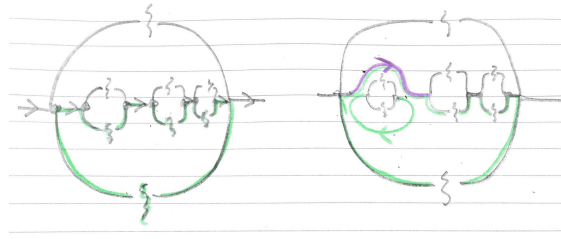
FIG. 6: Two faces which share more than one edge must create an opportunity for a cut and splice.

Since number of vertices and edges of $\mathfrak{L}\#_{e,e'}\mathfrak{L}'$ is the sum of the vertices and edges in $\mathfrak{L}$ and $\mathfrak{L}'$ and the number of faces is 2 less than the sum of the number of faces in $\mathfrak{L}$ and $\mathfrak{L}'$, $V - E + F = 2$ for $\mathfrak{L}\#_{e,e'}\mathfrak{L}'$, so the connect sum pd-code is planar as desired. □

We now give a theorem corresponding to the prime decomposition of links [19]:

Proposition 13. *Every composite link shadow can be created by connect sum operations on a well-defined set of prime link shadows called the prime factors of the composite shadow.*

Proof. We proceed by induction on the number of pairs of edges shared by two faces. If no two faces share more than one edge, the number of pairs is zero and the diagram is already prime. This is the base case.

Observe that as in Figure 6, if some pair of faces shares some number of edges, the shared edges occur in precisely opposite order on both faces (if not, the diagram isn't planar). This means that our diagram is the connect sum of two subdiagrams created by cutting and splicing a pair of adjacent shared edges, as on the right side of that Figure.

Since each subdiagram has fewer pairs of edges shared by two faces, by induction we may write them as connect sums of collections of prime diagrams. But we made arbitrary choices when deciding which pair of adjacent edges to cut-and-splice, and so must check that the collection of prime diagrams did not depend on these choices.

It's enough to show that the collection of subdiagrams produced by doing any two such cut-and-splices in one order is the same as the collection of subdiagrams produced by doing them in the other. But this is clear as the operations don't interfere with one another (Figure 7 illustrates the point.) □

We now know that every n vertex composite shadow L is the connect sum of prime shadows $L_1, \ldots, L_k$ and that the numbers of vertices $V_1, \ldots, V_k$ form an integer partition of n. Therefore, we need to enumerate "all connect sums of all link shadows with numbers of vertices that partition n". This is not as trivial as it sounds. Writing the connect sum as $L_1\#\ldots\#L_k$ is dangerously misleading– though connect sum is associative (and even commutative) on isotopy classes of knots,
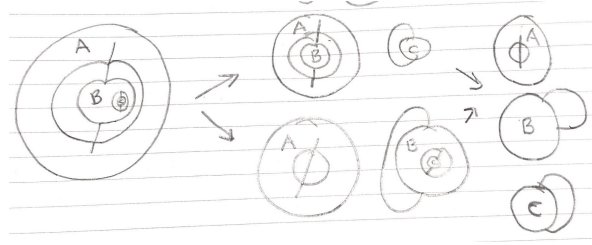
FIG. 7: Cutting and splicing at two locations can be done in either order and produce the same list of subdiagrams.
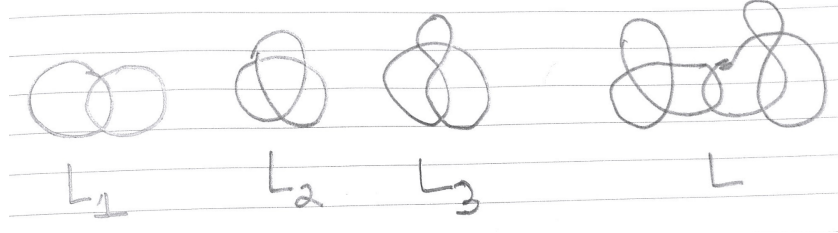

FIG. 8: L can be written as $L = (L_1 \# L_2) \# L_3$ but not as $L_1 \# (L_2 \# L_3)$.

the same is not true for links and for diagrams. Figure 8 gives an example of a link shadow L which is a connect sum of diagrams L_1, L_2 and L_3, which *can* be written $(L_1 \# L_2) \# L_3$ but *can't* be written $L_1 \# (L_2 \# L_3)$. It is easy to see the following.

Lemma 14. *The set of prime summands $\{L_1, \ldots, L_k\}$ of a composite link shadow L can be renumbered so that $L = (((L_1 \# L_2) \# L_3) \# \cdots L_k)$ and L_1 has the largest number of vertices among all the $L_1, \ldots, L_k$. The numbers of vertices $V_2, \ldots, V_k$ in the other prime summands are* not *guaranteed to be in sorted order.*

Proof. Since the overall shadow is connected, we can start connect-summing at any prime summand and build the rest of the shadow from there by undoing the cut-and-splice operations of Proposition 13. Therefore, we are free to choose L_1 to have a maximal number of vertices. $\square$

Lemma 15. *The set $\{L_1, \ldots, L_k\}$ of the prime summands of L is a shadow-isomorphism invariant of L. In particular, the set of numbers of vertices $\{V_1, \ldots, V_k\}$ is a shadow isomorphism invariant of L.*

Proof. This follows directly from that the set $\{L_1, \ldots, L_k\}$ is well-defined. $\square$

Now recall that a pdstor (Definition 9) is an ordered collection of canonical pd-codes which represent pd-isomorphism classes. We can define connect sum for pdstors by saying that $P_1 \# P_2$

is the pdstor constructed by adding all $\mathfrak{L}_1\#_{e1,e2}\mathfrak{L}_2$ where e_i is an edge of some $\mathfrak{L}_i \in P_i$. We now generate composite link shadows iteratively as shown in the pseudocode in Figure 9.

procedure BUILDCOMPOSITESHADOWS(n)   ▷ Build all link shadows with $\leq n$ vertices
    Use *plantri* to build pdstors $P_1, \ldots, P_n$ containing all oriented prime link shadows with 1 to n vertices.
    for all $1 \leq k \leq n$ **do**
        Define an empty pdstor P_k.
        for all partially sorted partitions $k_1 k_2 \ldots k_\ell$ of k with $k_1 \geq k_i$ **do**
            Build the pdstor $P_{k_1 k_2 \cdots k_\ell} = ((P_{k_1}\#P_{k_2})\#P_{k_3})\# \cdots P_{k_\ell}$.
            (If $((P_{k_1}\#P_{k_2})\#P_{k_3})\# \cdots P_{k_{\ell-1}}$ was already computed, we can reuse it.)
        end for
        for all *fully* sorted partitions $k_1 k_2 \ldots k_\ell$ of k with $k_1 \geq k_2 \geq \cdots \geq k_\ell$ **do**
            Define an empty pdstor P.
            for all *partially* sorted partitions $k_1' k_2' \cdots k_\ell'$ with the same *set* of k_i **do**
                Add all elements of $P_{k_1' k_2' \cdots k_\ell'}$ to P.
                (Some of these will be isomorphic to one another.)
            end for
            Add all elements of P to P_k.
            (By Lemma 15, these aren't isomorphic to anything previously computed,
            so we don't need to check for isomorphism with existing elements of P_k.)
        end for
    end for
end procedure

FIG. 9: Our algorithm for generating composite link shadows proceeds in several rounds to save computational time. Comparing the generating composites to the existing database to eliminate duplicates can consume a considerable amount of time even when using a hashing scheme.

3.2. Expansions of embedded planar simple graphs

Our next strategy will be much more complicated (and somewhat slower to run), but it serves as crucial check on the previous computation. The basic idea is to define a smaller class of graphs so that the graphs we are interested in can be obtained from the base class of graphs by various expansion moves. Lehel [23] gave a strategy for generating all 4-regular graphs in this way from the octahedral graph. Instead of using Lehel's strategy directly, we build on the method of Brinkmann and McKay [5, 26] for enumerating isomorph-free embedded planar graphs; we extend their work here to generate the class of graphs that we're interested in.

We observed above that the link shadows are embedded isomorphism classes of 4-regular embedded planar multigraphs. We now define four expansion moves of embedded planar graphs with vertex degree ≤ 4 which generate embedded planar multigraphs of vertex degree ≤ 4 with the same number of vertices, but additional edges:

Definition 16. The four expansion operations that we will use are the following:

- E_1 loop insertion adds a loop edge to a vertex of degree 1 or 2, as below. Loop insertion can be performed on each side of a vertex of degree 2.

FIG. 10: Adding a loop edge to a vertex of degree 1 or 2.

- E_2 reversing edge doubling duplicates an existing edge joining vertices of degree < 4 so as to create a new bigon face. Note that the (counterclockwise) order of the two vertices is reversed on the two vertices.
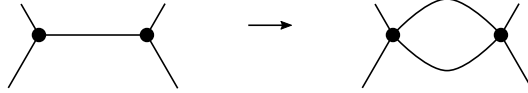
FIG. 11: Doubling an edge joining two vertices of degree < 4.

- E_3 preserving doubling also duplicates an existing edge joining vertices of degree < 4, but keeps the counterclockwise order of the edges the same on each at each of the two vertices. This sort of doubling is only available if the original edge is a cut edge of the graph.

FIG. 12: Doubling a cut edge joining two vertices of degree < 4 can be done another way.

- E_4 pair insertion adds a pair of edges simultaneously, joining two vertices of degree 2 which are both on two faces of the embedding, as below.
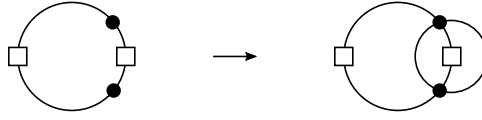
FIG. 13: Adding two new edges to join vertices of degree 2 on the same face.

We can now show

Proposition 17. *Every link shadow L can be obtained from a connected, embedded planar simple graph of vertex degree ≤ 4 G_0 by a series of E_1, E_2, E_3, and E_4 expansions.*

Equivalently, any link shadow L can be reduced to a connected embedded planar simple graph L_0 of vertex degree ≤ 4 by a series of E_1, E_2, E_3, and E_4 reductions. The embedded isomorphism type of L_0 is determined uniquely by the (unoriented) shadow isomorphism type of L (the order in which the reductions are performed doesn't matter).

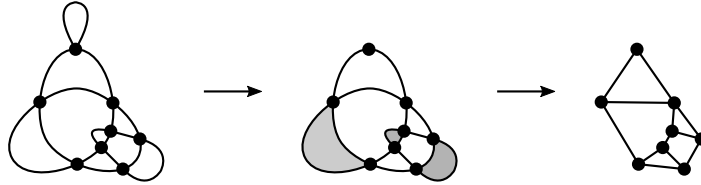An illustration of the process we describe is shown in Figure 14.

FIG. 14: Any link shadow can be reduced to a connected, embedded planar simple graph of vertex degree ≤ 4 by a series of reductions, according to Proposition 17.

The proof appears in Appendix B. We can now build the link shadows by applying expansions to the graphs produced by *plantri*. This is not trivial. First, not all sequences of expansion moves lead to link shadows. Second, different sequences of expansion moves may produce isomorphic link shadows. Some examples are shown in Figure 15.

We can think of this as the problem of searching for collections of expansion moves which obey various equations, such as the fact that the total vertex degree must be 4 in any complete solution (there are less obvious equations as well– the complete system is specified in Appendix C).

We find all of the solutions using a standard branch-and-bound algorithm and add the results to a pdstor to eliminate isomorphic pd-codes. We can save a lot of time in this process by noting that Proposition 17 tells us that we need only check expansions of the same graph (and their reorientations) against each other for potential pd-isomorphisms.
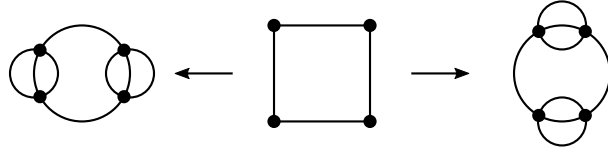
FIG. 15: Two different sequences of E_2 moves lead to isomorphic expansions of this simple graph. This means that the output of the expansion process will contain duplicate link shadows which must be detected and eliminated.

4.   COMPUTING AND VERIFYING THE RESULTS

We implemented the above algorithms in C and used them to generate a list of link shadows up to 10 crossings, each unique up to pd-isomorphism. The longest run took several days on a desktop computer. We then extracted the knot shadows, leaving the link shadows for future work. We assigned orientations and over-under information at the crossings to each shadow to make a list of diagrams containing various duplicate diagrams due to symmetry. We eliminated diagram-isomorphic duplicates from this list to arrive at a database of diagrams. Many knots were identified uniquely by their HOMFLY polynomial, which we computed using the Millett/Ewing HOMFLY-PT code [16]. About 6.5 million diagrams whose HOMFLY corresponded to more than one knot were classified by using Bar-Natan and Morrison's KnotTheory *Mathematica* package to compute the Kauffman polynomial and signature. Since KnotTheory gives incorrect answers for diagrams with one-component "loop" faces, we had to eliminate these faces before computation.

This part of the computation was large enough to require organization; several thousand hours of computer time were required to expand the 10 crossing shadows into diagrams and compute their HOMFLY polynomials. We first divided the 10 crossing shadows into roughly 10,000 pieces, each corresponding to about 20 minutes of computation time. These input files were stored on the Amazon S3 storage service. We then entered a message for each input file into a queue in the Amazon SQS service. Worker processors read job descriptions from the queue; SQS then placed a temporary hold on the messages. If the job completed, the workers deleted the job message from the queue and uploaded their results to S3; if jobs failed to complete, SQS placed the message back into the queue after a delay of one hour. The worker processes ran on a mixture of local hardware and virtual Linux machines running in Amazon computing centers in Oregon and Virginia. Time on the virtual machines was obtained by bidding for an hourly price for computation. We paid an average of 0.4 cents/CPU hour for virtual machines and were able to lease 400 simultaneous cores at this price. The counts of shadows and diagrams are given in Table I.

The most important question, of course, is how the computation was checked. Our implementation was careful and involved quite a bit of internal self-checking as well as testing against *valgrind*

Cr	Prime Shadows	Link Shadows	Knot Shadows	Knot Diagrams
3	1	7	6	36*
4	2	30	19	276*
5	3	124	76	2936*
6	9	733	376	35 872*
7	18	4586	2194	484 088*
8	62	33 373	14 614	6 967 942*
9	198	259 434*	106 421	105 555 336*
10	803	2 152 298*	823 832	1 664 142 836*

TABLE I: The number of link (including knots) and knot shadows and diagrams through 10 crossings. The unstarred numbers in the column of prime shadows are sequence A113201 in the OEIS. The unstarred numbers in the table of link shadows matche Kápolnai et al. [21]. The numbers in the column of knot shadows are sequence A008989 in the OEIS. They match the "UU, $g = 0$" row on page 43 of the the recent preprint of Coquereaux et al. [9], including the value 823 832 for Cr = 10 which those authors give as "should be confirmed". The starred numbers are new.

for memory problems, and writing a suite of unit tests for the codebase. However, good programming practices can only provide a limited measure of confidence in the results, so we continued to test our work. The first and most important test was to verify that the lists of knot and link shadows obtained by connect summing and by expansions were identical.

We were also able to check against some existing enumerations. Kápolnai et al. classified spherical "multiquadrangulations" [21], which are the duals of our diagrams as explained in Section 3.1. Their table 2 of the count of multiquadrangulations matches our count of diagrams exactly through 8 crossings (note that the quadrangulation has two more vertices than the dual knot diagram has crossings, so their data is shifted by two). It's worth observing that these authors also use *plantri*, so their results are not completely independent from ours. Still, it provides some comfort to see that their implementation on top of *plantri* produces the same results as ours. For knot shadows in particular, Arnol'd has given counts of the number of immersions of the unoriented circle into the unoriented sphere with n crossings for n from 0 to 5 [3, page 79]. This is sequence A008989 in the Online Encyclopedia of Integer Sequences, with an extension to $n = 7$ credited to Guy H. Valette. We could not find a published reference for Vallette's extension of the table, but our data for knots does match A008989 including the extension. Coquereaux et al. [9] have recently extended Valette's count using other means (their count is independent of *plantri*) and our numbers also match those in that paper.

Looking at Table I, one is struck by how close the number of distinct knot diagrams is to the maximum number $2^{Cr+1} \times$ (# knot shadows). To take 8 crossing diagrams as an example, we would expect at most $7\,482\,368 = 14\,614 \times 2^9$, and we have 6 967 942– about 93% of the maximum possible number. By the time we reach 10 crossing diagrams, the corresponding fraction is roughly 98.6%. We have fewer distinct diagrams only because some of the underlying knot diagrams have symmetries. For instance, the trefoil diagram 3_1 has a 3-fold rotational symmetry, so the crossing

Cr	3	4	5	6	7	8	9	10
Mean Automorphisms	5	$\frac{64}{19}$	$\frac{44}{19}$	$\frac{159}{94}$	$\frac{1447}{1097}$	$\frac{8426}{7307}$	$\frac{113460}{106421}$	$\frac{846979}{823832}$
— (decimal)	5.00	3.37	2.32	1.69	1.32	1.15	1.07	1.03

TABLE II: The mean number of automorphisms decreases rapidly as the number of crossings in the diagram increases. This means that the number of knot diagrams (with crossing signs and orientations) rapidly approaches the maximum allowed by the number of knot shadows.

sign assignments $+--$, $-+-$ and $--+$ are all the same. However, our computations reveal that such symmetries quickly become very rare as the number of crossings increases. Table II shows the mean number of automorphisms of a knot shadow, which rapidly decreases to 1 (the identity map), as is known for sufficiently large numbers of crossings [7].

4.1. Knot Types and Inferred Counts

Counting knot types required us to be careful about knot symmetries. To review, mirroring crossings and reversing orientation yield a $\mathbb{Z}_2 \times \mathbb{Z}_2$ action on knot types. If a knot is isotopic to its image under a subgroup of this group, it is said to have a symmetry; there are five symmetry types corresponding to the five subgroups of this group: "none", "mirror", "reversible", "amphichiral" (the diagonal subgroup), and "full". We refer to a collection of knot types related by the group as a "base knot type". For instance, the base knot type 3_1 consists of the two knot types 3_1 and 3_1^m (the mirror image of 3_1).

We were able to use the HOMFLY-PT and Kauffmann polynomials and the knot signature to classify almost all of the knots whose base type had symmetry "full" or "reversible", since these invariants distinguish all knots with 10 crossings or less from their mirror images except the 10_{71} knot. There are 36 base types with one of the other three symmetries. These were more difficult to classify as classical invariants don't distinguish knots from their reversals. We relied on symmetry to infer the distribution of counts:

Lemma 18. *If a base knot type K has symmetry type "amphichiral" ($K = K^{mr}$, $K^m = K^r$, but $K \neq K^r$) or "mirror" ($K = K^m$, $K^r = K^{mr}$, but $K \neq K^r$), then the number of diagrams of the two knot types are equal. If K has symmetry type "none" ($K \neq K^m \neq K^r \neq K^{mr}$) the number of diagrams of each of these four knot types are equal.*

Proof. In each case, there is a group action on diagrams (reverse the orientation of a diagram or take the 4-element group generated by reversing crossing signs and orientation) which converts diagrams of one knot of these knot types to a diagrams of the other types. This action extends to diagram-isomorphism classes of diagrams because the action commutes with any given diagram-isomorphism.

It is not always the case that this action is free on diagram-isomorphism classes of diagrams of any knot type. For instance, taking the mirror image of this diagram of the 7_4 knot ⬡ is equivalent to rotating it by 180 degrees (a diagram-isomorphism). This implies that 7_4 is a reversible knot, as diagram-isomorphisms are knot isotopies. But by hypothesis, our K is *not* isotopic to its images under the group action, and therefore it cannot by diagram-isomorphic either.

Since the action of the group on this set of diagram-isomorphism classes is free and exchanges the various knot types, the number of diagram-isomorphism classes of diagrams in each knot type must be the same. □

For example, using this lemma we split our original count of 5672 10-crossing diagrams of base type 8_{17} into 2836 diagrams of type 8_{17} and 2836 diagrams of type 8_{17}^r, even though we had no way of knowing which of our diagrams was assigned to each knot type.

The relative frequencies of all the knot types appear in Figure 16 in a log-log plot. The figure also shows the 10 most frequent knot types, from the unknot (the most frequent knot type in all our data) and the trefoils 3_1 and 3_1^m through the square knot $3_1 \# 3_1^m$ and the 6_3 knot, which is the 10th most common knot in all our data. Though we know [7] both that unknotted diagrams eventually become exponentially rare, and that the rank-order of the knot types must eventually change, we do not see this effect in our data; the list of 10 most common knots is the same for crossing numbers 6-10. Table III gives more detailed frequency data for the 40 most common knot types (among 10 crossing diagrams). This data does show some reordering of knot types as crossing number increases.

More interestingly, the log-log plot of ranked knot frequencies for the 622 different possible knot types for 10 crossing diagrams is roughly linear over 9 orders of magnitude in frequency (there are about 1.6×10^9 diagrams, with the most frequent knot type (the unknot) occurring 1.2×10^9 times and the least frequent knot types (a 98-way tie among various 10-crossing knots) appearing exactly once. It is not clear to us why this phenomenon should occur in the data: this data is certainly compatible with the hypothesis that there is an (asymptotic) power-law relationship between knot rank and knot probability akin to Zipf's law, but it would take much larger experiments to provide strong statistical support for such a conjecture.

At right in Figure 16 we give the unknot fraction explicitly in tabular form. We now try to understand this fraction. Of course, for any shadow we can set the crossings to produce an unknot by going "downhill" from some distinguished edge, but this effect can only account for a tiny fraction of the unknots– there are at most n ways to assign crossings in this manner but there are 2^n different crossing assignments overall. It turns out, however, that structural properties of the set of shadows explain most of the unknot fraction. To see how, we start with Figure 17, which shows the first 42 shadows obtained in the enumeration. It is immediately clear from Figure 17 that many of the shadows can be simplified by a Reidemeister I move; they have a face with only one edge (a monogon). In fact, we can see from Table IV that almost every diagram has at least one monogon and that the mean number of monogons seems to be rising linearly with n. It is

Crossing Number	3	4	5	6	7	8	9	10
Number of Diagrams	36	276	2936	35 872	484 088	6 967 942	105 555 336	1 664 142 836
0_1	34	265	2744	32 456	422 332	5 852 832	85 253 534	1 291 291 155
3_1	1	5	85	1466	25 432	440 570	7 696 083	135 702 456
3_1^m	1	5	85	1466	25 432	440 570	7 696 083	135 702 456
4_1	–	1	18	412	8450	165 791	3 175 612	60 146 706
5_2	–	–	1	24	730	18 075	415 290	9 025 926
5_2^m	–	–	1	24	730	18 075	415 290	9 025 926
$3_1\#3_1^m$	–	–	–	2	112	3953	113 684	2 923 783
6_3	–	–	–	2	106	3515	96 666	2 389 180
6_2	–	–	–	1	58	2027	58 354	1 493 624
6_2^m	–	–	–	1	58	2027	58 354	1 493 624
$3_1\#3_1$	–	–	–	2	58	2006	56 893	1 461 498
$3_1^m\#3_1^m$	–	–	–	2	58	2006	56 893	1 461 498
6_1^m	–	–	–	1	34	1267	38 199	1 015 996
6_1	–	–	–	1	34	1267	38 199	1 015 996
$3_1\#4_1$	–	–	–	–	8	516	20 458	648 362
$3_1^m\#4_1$	–	–	–	–	8	516	20 458	648 362
7_6^m	–	–	–	–	3	193	7608	240 121
7_6	–	–	–	–	3	193	7608	240 121
7_7	–	–	–	–	2	124	4709	144 455
7_7^m	–	–	–	–	2	124	4709	144 455
7_5^m	–	–	–	–	2	102	4244	138 467
7_5	–	–	–	–	2	102	4244	138 467
7_2^m	–	–	–	–	1	44	2103	74 739
7_2	–	–	–	–	1	44	2103	74 739
7_3	–	–	–	–	1	39	1793	62 059
7_3^m	–	–	–	–	1	39	1793	62 059
$4_1\#4_1$	–	–	–	–	–	20	1176	51 526
7_4^m	–	–	–	–	1	36	1516	49 731
7_4	–	–	–	–	1	36	1516	49 731
8_{20}	–	–	–	–	–	14	985	41 843
8_{20}^m	–	–	–	–	–	14	985	41 843
$3_1^m\#5_2^m$	–	–	–	–	–	10	784	36 548
$3_1\#5_2$	–	–	–	–	–	10	784	36 548
$3_1^m\#5_2$	–	–	–	–	–	10	784	36 544
$3_1\#5_2^m$	–	–	–	–	–	10	784	36 544
8_{21}	–	–	–	–	–	9	574	24 611
8_{21}^m	–	–	–	–	–	9	574	24 611
8_{14}	–	–	–	–	–	6	442	19 412
8_{14}^m	–	–	–	–	–	6	442	19 412
7_1^m	–	–	–	–	1	8	444	17 441

TABLE III: This table shows the number of diagrams of each knot type among all knot diagrams with between 3 and 10 crossings. The knot types shown are the most common 40 knot types among 10 crossing diagrams, and they appear in the order of their frequency among 10 crossing diagrams. This is not the same rank order for all crossing numbers– one can observe that $4_1\#4_1$ is less common than 7_4 among 8 crossing diagrams (20 diagrams versus 36 diagrams) but more common than 7_4 among 10 crossing diagrams (51 526 diagrams versus 49 731 diagrams).
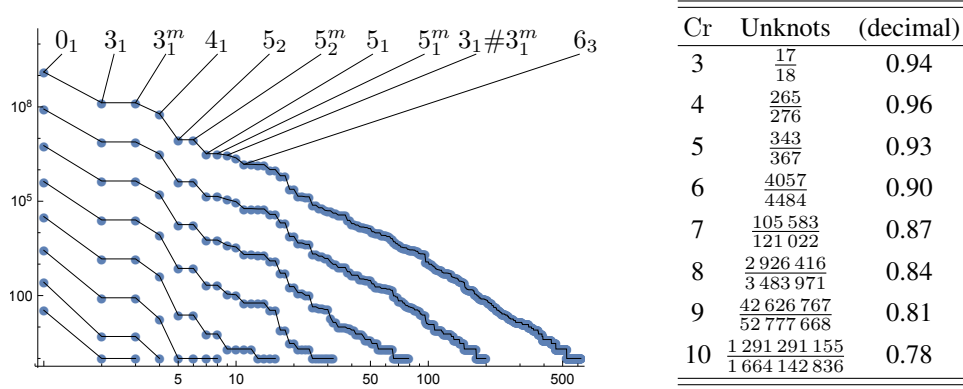
0_1 3_1 3_1^m 4_1 5_2 5_2^m 5_1 5_1^m $3_1\#3_1^m$ 6_3

Cr	Unknots	(decimal)
3	$\frac{17}{18}$	0.94
4	$\frac{265}{276}$	0.96
5	$\frac{343}{367}$	0.93
6	$\frac{4057}{4484}$	0.90
7	$\frac{105\,583}{121\,022}$	0.87
8	$\frac{2\,926\,416}{3\,483\,971}$	0.84
9	$\frac{42\,626\,767}{52\,777\,668}$	0.81
10	$\frac{1\,291\,291\,155}{1\,664\,142\,836}$	0.78

FIG. 16: A log-log plot of knot frequencies in rank order for crossing numbers 3 through 10, with the 10 most common knot types identified. These knot types have the same rank ordering in all the crossing numbers computed. The data show some evidence of power-law behavior. At right, we see the unknot fraction in table form.

Cr	3	4	5	6	7	8	9	10
Mean # monogons	$\frac{12}{6}$	$\frac{48}{19}$	$\frac{213}{76}$	$\frac{1196}{376}$	$\frac{7714}{2194}$	$\frac{56\,540}{14\,614}$	$\frac{448\,584}{106\,421}$	$\frac{3\,758\,456}{823\,832}$
— (decimal)	2.	2.53	2.8	3.18	3.52	3.87	4.22	4.56
Monogon fraction	$\frac{5}{6}$	$\frac{18}{19}$	$\frac{74}{76}$	$\frac{371}{376}$	$\frac{2178}{2194}$	$\frac{14\,562}{14\,614}$	$\frac{106\,216}{106\,421}$	$\frac{822\,989}{823\,832}$
— (decimal)	0.833	0.947	0.974	0.987	0.993	0.996	0.998	0.999
Mean # bigons	$\frac{6}{6}$	$\frac{18}{19}$	$\frac{88}{76}$	$\frac{470}{376}$	$\frac{3037}{2194}$	$\frac{21\,925}{14\,614}$	$\frac{173\,342}{106\,421}$	$\frac{1\,450\,209}{823\,832}$
— (decimal)	1.	0.947	1.16	1.25	1.38	1.5	1.63	1.76
Bigon fraction	$\frac{3}{6}$	$\frac{11}{19}$	$\frac{52}{76}$	$\frac{275}{376}$	$\frac{1714}{2194}$	$\frac{11\,892}{14\,614}$	$\frac{89\,627}{106\,421}$	$\frac{712\,961}{823\,832}$
— (decimal)	0.5	0.579	0.684	0.731	0.781	0.814	0.842	0.865

TABLE IV: The distribution of monogons among shadows of n crossings. The mean number of monogons in an n-crossing diagram fits very well to $1.01927 + 0.356111n$, while the mean number of bigons fits well to $0.563245 + 0.117749n$ It would be interesting to know the asymptotic growth rate.

interesting to note that almost every diagram contains either a monogon or a bigon– there are only three diagrams in our dataset without one or the other! These turn out to be the shadows of torus knots and links that Conway designated as 8^*, 9^*, and 10^*. This means that we can expect to reduce the complexity of an average shadow substantially just by eliminating monogons, which can be done regardless of crossing signs. We note here that there are some adjustments to our enumeration scheme which may produce an interesting subset of shadows, but do not provide any proof: If the E_1 loop insertion rule is removed from the graph expansion procedure, all shadows without monogons should be tabulated. Similarly, removing the one-crossing ∞ diagram from our connect-summing procedure should tabulate all diagrams without nugatory crossings.
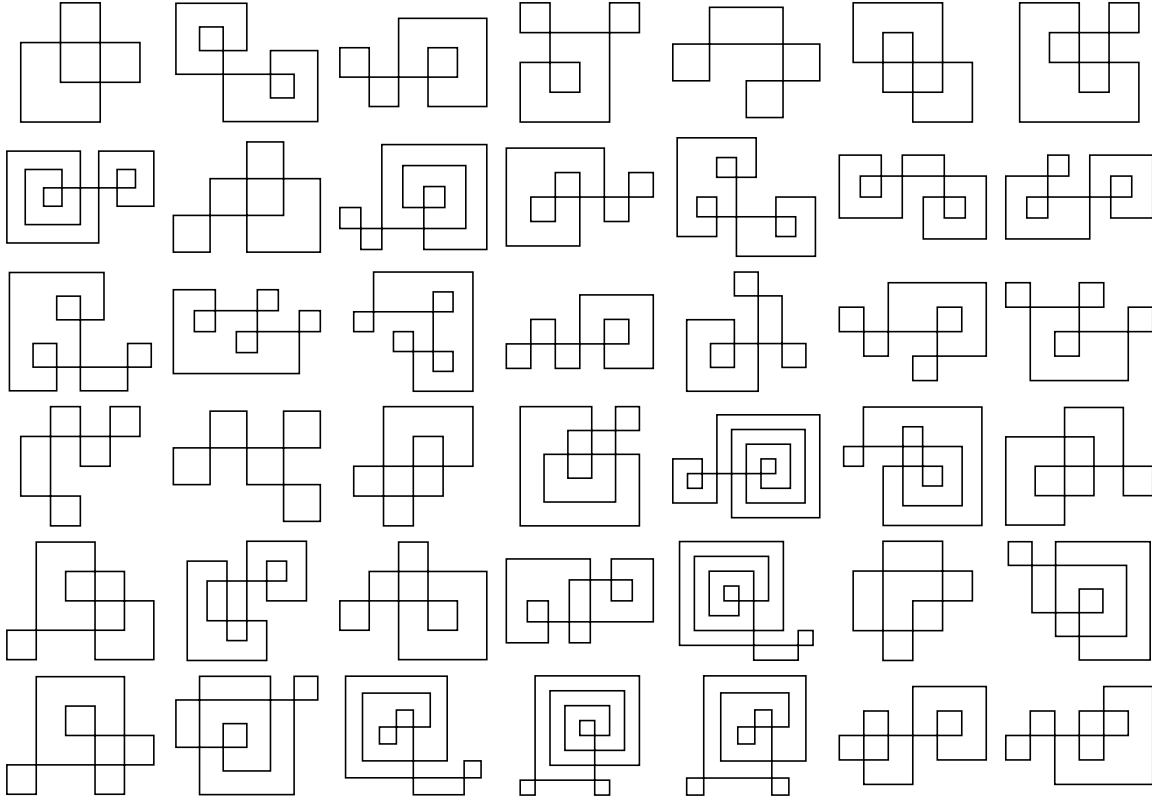
FIG. 17: The first 42 knot shadows.

Put another way, almost all knot shadows are composite with the one-crossing diagram $\bigcirc\!\bigcirc$ as a (diagrammatically) prime factor. Figure 18 shows the fraction of shadows of a given crossing number with a given number of prime summands. If the number of prime summands is equal to the crossing number n, the shadow is a connect sum of one-crossing 8 diagrams– these diagrams are called "tree-like" by Aicardi [2]. Tree-like diagrams are colored in dark on the left-hand side of Figure 18. There are no n-crossing shadows with $n-1$ prime summands (as there are no 2-crossing prime knot diagrams).

We will call the n-crossing shadows with $n-2$ prime summands "3-almost tree-like" as they are the connect sum of a tree-like diagram and a 3-crossing prime diagram, and similarly call the n-crossing diagrams with $n-3$ prime summands "4-almost tree-like" as they are the connect sum of a tree-like diagram and a 4-crossing prime diagram. These diagrams are colored in on Figure 18 in a lighter color.

Proposition 19. *If a shadow is tree-like, all assignments of crossings result in the unknot; these are the only knot shadows with this property. If a shadow is 3-almost tree-like, $3/4$ of crossing assignments produce unknots. If a shadow is 4-almost tree-like, $7/8$ of crossing assignments produce*
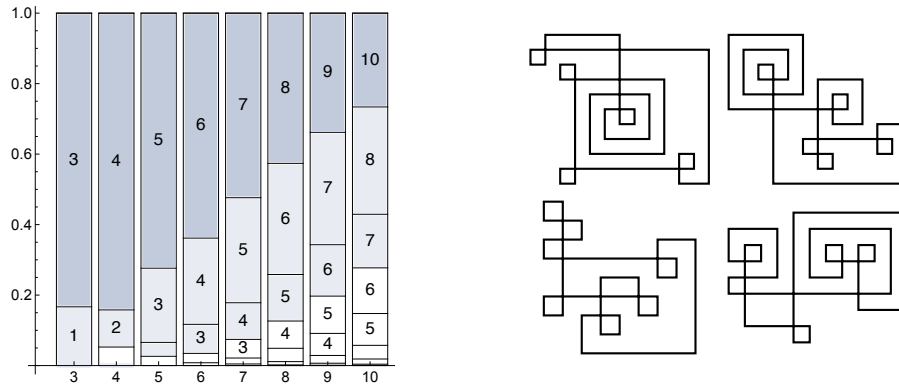
FIG. 18: The chart at left shows the (proportional) breakdown of shadows with 3 through 10 crossings into collections of shadows with the same number of prime summands. We see in the far left rectangle, for instance, that more than 80% of 3-crossing shadows are composed of connect sums of 3 one-crossing 8 diagrams (these are colored dark-gray and labelled '3'), while the far right rectangle shows only about 25% of 10-crossing diagrams are connect sums of 10 one-crossing 8 diagrams. The diagrams where the number of crossings and summands are equal are called tree-like (cf. [2])– they are always unknotted regardless of crossing assignments, and are shaded dark grey throughout. At right, we see several tree-like shadows with 8 crossings.

unknots.

Proof. The calculations of unknot fractions are easy; a tree-like diagram can be reduced to the unknot by Reidemeister I moves regardless of crossing assignment. Further, there is only one prime knot shadow of 3 or 4 crossings; the assignment of all $+$ or all $-$ crossings yields a 3_1 or 4_1 knot, while all other assignments result in unknots.

The statement that the tree-like shadows are the only shadows with the property that all crossing assignments produce unknots is more interesting. Polyak [32] shows that the average value of the Vassiliev v_2 invariant (which is 0 for unknots) over all crossing assignments for a shadow is given by the formula $\frac{1}{8}(\mathrm{J}^+ + 2\,\mathrm{St})$ where J^+ and St are Arnol'd invariants of the shadow (cf. [3]). (While J^+ and St are not spherically invariant, the sum $\mathrm{J}^+ + 2\,\mathrm{St}$ is.) Further, both he and Aicardi [2] show that tree-like shadows are the only shadows with $\mathrm{J}^+ + 2\,\mathrm{St} = 0$. $\square$

Doing the requisite sums, this analysis predicts that roughly 77% of 8-crossing diagrams, 70% of 9-crossing diagrams, and 63% of 10-crossing diagrams should be tree-like or almost tree-like unknots. We can compare these figures to the actual unknot fractions from Figure 16: roughly 83% of 8-crossing diagrams, 80% of 9-crossing diagrams, and 78% of 10-crossing diagrams are unknots of any kind. This suggests that the treelike phenomenon explains much of the unknot fraction. We note that these figures are not entirely comparable; while the Proposition shows that at least 63% of crossing assignments to 10-crossing shadows result in unknots, some of these crossing assignments might result in diagram-isomorphic diagrams if the underlying shadow has a symmetry, and so the

unknot fraction could be slightly smaller. This cannot make much difference to the estimate. The counts in Table I show that about 98.6% of crossing assignments to 10 crossing shadows produce distinct diagrams, so the worst-case scenario is that this analysis shows that at least 61.4% of 10-crossing diagrams are tree-like or almost tree-like unknots.

5.   FUTURE DIRECTIONS

All of collections of knot shadows are included as supplementary data with this paper, including files of coordinates for planar embeddings of the shadows as well as their pd-codes. In addition, we include the actual counts of diagrams of various knot types as CSV files, and the files of diagrams with 8 crossings and fewer. We did not assemble files of diagrams for 9 and 10 crossing diagrams, but classified diagrams on-the-fly as we generated them (we estimate the file of 10-crossing diagrams to be about 1.6 tb in size if generated). Our source code is also available publicly.

Once in possession of an enumeration of diagrams, it is tempting to compute knot distances (cf. [10, 28]) and unknotting numbers. We have carried out some preliminary experiments along these lines and are disappointed to report that it does not seem to resolve any of the remaining uncertainties in the knot distance tables.

More sophisticated counting procedures have been applied in recent years, particularly for shadows of knots. Jacobsen and Zinn-Justin[20] give a transfer matrix analysis for a combinatorial description of prime, reduced, weighted, marked planar shadows for knots very similar to Gusein-Zade's enumeration [17, 18]. Gusein-Zade's actual computer enumeration was carried out only to 10 crossings while Jacobsen and Zinn-Justin push the enumeration to the nearly unthinkable 22-crossing case, where they give an exact count of 40558226664529044000 knot shadows. Of course, their enumeration does not construct each shadow, so it cannot be used easily to estimate the unknot fraction.

Schaeffer [33] (see also [4]) constructed a very insightful bijection between link shadows with a distinguished, directed edge ("rooted link shadows") and a class of decorated trees called "blossom trees". One of us (Chapman), uses this identification to analyze the asymptotic behavior of our model in further work [7], proving that most diagrams are heavily composite and hence represent nontrivial knots.

Acknowledgments

Appendix A: pd-codes and isomorphisms

In this appendix, we give more detail on our algorithm for detecting pd-isomorphic shadows. It is clear that a lot of data about a pd-code is preserved by isomorphism: for instance, the number of crossings, edges, faces, and the numbers of edges around faces. We can use this information to rule out isomorphisms using a hashing scheme.

Definition 20. Suppose the pd-code P has V crossings, E edges, F faces, and C components. We assume that the faces are denoted $f_1, \ldots, f_F$ and the components are denoted $c_1, \ldots, c_C$. Further, let $\mathrm{edges}(x)$ give the number of edges on a face or component. Then the *hash* of P is given by the tuple

$$\mathcal{H}(P) = (V, E, F, C, \{\mathrm{edges}(f_1), \ldots, \mathrm{edges}(f_F)\}, \{\mathrm{edges}(c_1), \ldots, \mathrm{edges}(c_C)\}).$$

The last two are unordered sets of integers.

It is clear that

Lemma 21. *If two pd-codes P_1 and P_2 are isomorphic, then $\mathcal{H}(P_1) = \mathcal{H}(P_2)$.*

Proof. The numbers V, E, F, and C are clearly preserved by isomorphism. The indices of edges and faces may be permuted by an isomorphism, but the number of edges on each can't change. Thus the *unordered sets* of edge counts for faces and component remain the same as well. $\square$

We can now build up an isomorphism between pdcodes by a series of definitions:

Definition 22. Suppose we have two pd-codes $\mathfrak{L}$ and $\mathfrak{L}'$ with the same hash.

- A bijection $\gamma : \{c_1, \ldots, c_C\} \to \{c'_1, \ldots, c'_C\}$ between the components of $\mathfrak{L}$ and the components of $\mathfrak{L}'$ is called *component-length preserving* if $\#\,\mathrm{edges}(c_i) = \#\,\mathrm{edges}(\gamma(c_i))$ for all i.

- Given such a component-length preserving bijection γ, a bijection $\epsilon : \{e_1, \ldots, e_E\} \to \{e'_1, \ldots, e'_E\}$ between the edges of $\mathfrak{L}$ and the edges of $\mathfrak{L}'$ is called *component-preserving* and *compatible with* γ if ϵ maps the edges of each c_i to the edges of $\gamma(c_i)$ by an element of the dihedral group $D_{\mathrm{edges}(c_i)}$. That is, the edges of c_i are mapped in cyclic (or reverse-cyclic) order to the corresponding edges of $\gamma(c_i)$.

- Given a component-preserving bijection $\epsilon : \{e_1, \ldots, e_E\} \to \{e'_1, \ldots, e'_E\}$ between the edges of $\mathfrak{L}$ and the edges of $\mathfrak{L}'$, we say that a bijection $\nu : \{v_1, \ldots, v_V\} \to \{v'_1, \ldots, v'_V\}$ between the vertices of $\mathfrak{L}$ and the vertices of $\mathfrak{L}'$ is *compatible with* ϵ if

$$\nu(\mathrm{head}(e_i)) = \mathrm{head}(\epsilon(e_i)) \quad \text{and} \quad \nu(\mathrm{tail}(e_i)) = \mathrm{tail}(\epsilon(e_i))$$

  when e_i is part of a component mapped by an orientation-preserving element of the dihedral group, and

$$\nu(\mathrm{head}(e_i)) = \mathrm{tail}(\epsilon(e_i)) \quad \text{and} \quad \nu(\mathrm{tail}(e_i)) = \mathrm{head}(\epsilon(e_i))$$

  when e_i is part of a component mapped by an orientation-reversing element of the dihedral group.

- Given γ, ϵ, and ν that obey all the above conditions, we say that they are:

  - *globally orientation-preserving* if the set of edges e_i, e_j, e_k, e_l incident to each vertex v of $\mathfrak{L}$ (in counterclockwise cyclic order) is mapped to the set of edges $\epsilon(e_i), \epsilon(e_j), \epsilon(e_k), \epsilon(e_l)$ incident to $\nu(v)$ in counterclockwise cyclic order.

  - *globally orientation reversing* if the $\epsilon(e_i), \epsilon(e_j), \epsilon(e_k), \epsilon(e_l)$ are incident to $\nu(v)$ but in clockwise cyclic order (for each v),

  - otherwise, the triple is *inconsistent*.

We then have

Proposition 23. *Given a pair of pd-codes $\mathfrak{L}$ and $\mathfrak{L}'$ with the same hash, each isomorphism of $\mathfrak{L}$ to $\mathfrak{L}'$ is given by a set of bijections γ between their components, ϵ between their edges, and ν between their vertices where γ is component-length preserving, ϵ is component-preserving and compatible with γ, ν is compatible with ϵ and the triple is globally orientation preserving or reversing (not inconsistent).*

Proof. This is the definition of pd-isomorphic, restated using Definition 22. $\qquad\square$

A few other observations are helpful:

Lemma 24. *If $\mathcal{H}(\mathfrak{L}) = \mathcal{H}(\mathfrak{L}')$ for pdcodes $\mathfrak{L}$ and $\mathfrak{L}'$, there is at least one component-length preserving $\gamma : \{c_1, \ldots, c_C\} \to \{c'_1, \ldots, c'_C\}$. All component-length preserving γ can be generated by iterating over a product of permutation groups.*

Proof. We can partition the components of $\mathfrak{L}$ (and $\mathfrak{L}'$) into subsets by component length. Since the hashes $\mathcal{H}(\mathfrak{L})$ and $\mathcal{H}(\mathfrak{L}')$ match, these partitions correspond for P and P'. The component-length preserving γ are exactly those bijections which respect this partition: we can construct all such γ by taking a product of permutation groups whose orders are the size of the subsets. At worst, all subsets have size 1 and there is a single such γ. $\qquad\square$

Lemma 25. *Given a component-length preserving γ and component-preserving and compatible ϵ, and a set of orientations for the components of $\mathfrak{L}$, there is at most one $\nu : \{v_1, \ldots, v_V\} \rightarrow \{v'_1, \ldots, v'_V\}$ which is compatible with ϵ and consistently oriented on components and we can construct ν as below.*

Proof. Each vertex v of $\mathfrak{L}$ is incident to four edges e_i, e_j, e_k, e_l. Without loss of generality, assume that $v = \text{tail}(e_i), \text{tail}(e_j), \text{head}(e_k)$ and $\text{head}(e_l)$. Then if ν is compatible with ϵ, we must have

$$\nu(v) = \text{tail}(\epsilon(e_i)) = \text{tail}(\epsilon(e_j)) = \text{head}(\epsilon(e_k)) = \text{head}(\epsilon(e_l)).$$

If the four terms on the right are equal, this defines $\nu(v)$. If not, there is no compatible ν. $\qquad\square$

We can now find all isomorphisms between two pdcodes computationally by a simple brute-force strategy given by the pseudocode in Figure 19.

procedure BUILDISOMORPHISMS($\mathfrak{L},\mathfrak{L}'$)   ▷ Build isomorphisms between pdcodes $\mathfrak{L}$ and $\mathfrak{L}'$
    if the hashes $\mathcal{H}(\mathfrak{L})$ and $\mathcal{H}(\mathfrak{L}')$ are different **then**
        P and P' are not isomorphic. Return $\emptyset$.
    end if
    for all component-length preserving $\gamma : \{c_1, \ldots, c_C\} \rightarrow \{c'_1, \ldots, c'_C\}$ **do**
        for all compatible and component-preserving ϵ **do**
            if a compatible ν exists **then**
                if ν is globally orientation preserving or reversing **then**
                    ϵ, ν define an isomorphism $\mathfrak{L} \rightarrow \mathfrak{L}'$
                end if
            end if
        end for
    end for
end procedure

FIG. 19: The code above outlines the algorithm for detecting pd-isomorphism between pd-codes. Aside from hashing, this is a simple brute-force search. It should be clear that there is a tradeoff between potentially more efficient ways to organize the computation (as do more general graph-isomorphism checkers like *nauty* [27]) and the likelihood of introducing bugs into the code. This algorithm emphasizes simplicity .

Appendix B: Proving Proposition 17

In this section, we give the proof of

Proposition 17. *Every link shadow L can be obtained from a connected, embedded planar simple graph of vertex degree ≤ 4 G_0 by a series of E_1, E_2, E_3, and E_4 expansions.*
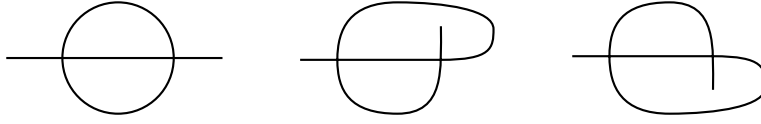
FIG. 20: By parity, because we came from a 4-regular embedded planar graph only the leftmost case can occur at any stage in the reduction process.

Equivalently, any link shadow L can be reduced to a connected embedded planar simple graph L_0 of vertex degree ≤ 4 by a series of E_1, E_2, E_3, and E_4 reductions. The embedded isomorphism type of L_0 is determined uniquely by the (unoriented) shadow isomorphism type of L (the order in which the reductions are performed doesn't matter).

Proof. We will prove the second statement, reducing in stages from some $G_n = G$ to G_0 by performing one reduction at each step. The number of steps we can perform is clearly finite, since each reduces the number of edges by at least one. So suppose we are at stage G_i. If there are no loop or multiple edges, we're done, and this is the simple graph G_0.

If there is a loop edge, we can remove it with a E_1 move.

If there is a multiple edge, we must consider several cases. We can think of each vertex of G_i as retaining a list of 4 connection points, ordered counterclockwise, from the initial embedding of G. Since we have performed some reductions already, some of these may be empty, but at least two are filled at each end of the multiple edge. Pick one vertex of the multiple edge and call it v and the other vertex w.
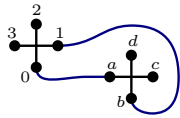
If the edge multiplicity is four, G is ⦶. This is obtained from the graph with one edge and two vertices by three E_2 moves.

If the edge multiplicity is three or two, there is at least one connection point on v which is not occupied by a copy of the multiple edge followed immediately by a connection point which is occupied by a copy e of the multiple edge. Without loss of generality, we'll call e the *base copy* of the multiple edge, and its connection point to at v position 0 around v. The remaining connection points will be numbered 1, 2, and 3. By construction, the edge joined to v at position 3 (if any) is not connected to w. We can label the other end of the base copy e position a on the second vertex w, and label the other positions b, c, and d, again counterclockwise.
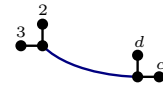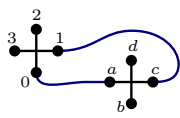
If the edge multiplicity is three, only one of these positions is unoccupied by a copy of the multiple edge. Looking at the three cases (shown in Figure 20), we can see that by parity, it must be position b, and the pair of copies $0a$ and $2c$ of the multiple edge can be removed by a E_4 operation. We have now disposed of the case where edge multiplicity is three.
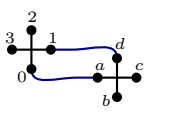
If edge multiplicity is two, there is one edge unaccounted for, which joins either position 1 or 2 on vertex v to position b, c, or d on vertex w. Therefore, there are six cases to address. We consider
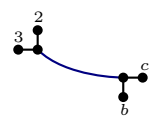
them in order, starting with the $1x$ configurations.

In the $1b$ configuration, the multiple edge forms a 2-cycle dividing the portion of the graph G connected to cd from the portion connected to 23. Deleting $1b$ requires a E_3 move, and the remaining base edge is a cut edge of all further-reduced G_i, as shown at right.
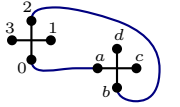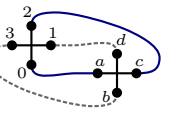
The $1c$ configuration is forbidden by parity.

In the $1d$ configuration, the multiple edge forms a bigon face. Deleting $1d$ uses an E_2 reduction, and yields the configuration at right. The remaining base edge may or may not be a cut edge of the further G_i.
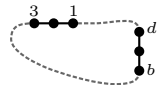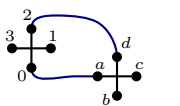
One might think that the $2-$ configurations are simply rearrangements of those above, but this is not true. A genuinely new case arises for $2c$.

The $2b$ configuration is forbidden by parity.

In the $2c$ configuration, by parity, the graph G must have connected 1 and d and also 3 and b. None of our moves change the connectivity of the graph (because we never delete all copies of a multiple edge), so the current graph G_i still joins these pairs of connection points. This means that we are in position for an E_4 pair reduction, resulting in the graph at right.

The $2d$ configuration is forbidden by parity.

Along the way, our analysis has been entirely local: we need only consider a single vertex to decide whether we can apply an E_1 reduction and a pair of vertices to decide on E_2, E_3, and E_4 operations. To show that order of operations doesn't matter, we need to show that whether or not we can apply

these operations does not depend on which reductions have already been performed. First, we note that since we never remove all copies of a multiple edge, we never change the connectivity of the graph during the reduction process.

The three copies of a multiplicity three edge must bound two bigons, and this does not change as we reduce other edges. Therefore, the E_4 move is always available for all multiplicity three edges.

Whether a multiplicity two edge is eligible for an E_2 move depends only on the positions of the ends of the multiple copies on their vertices, which don't change as we reduce. Therefore, this operation can always be performed (or is always forbidden), regardless of which reductions have already been performed.

Whether a multiplicity two edge is eligible for a E_3 or E_4 operation depends not only on the positions of ends of edges on their vertices, but also on the connectivity of the (reduced) graph. However, as we noted above, the connectivity of the graph doesn't change as we perform reductions.

It is clear that the isomorphism type of G_0 does not depend on the order of reduction– after all, in the end we are simply reducing the multiplicity of multiple edges of the graph.

It takes only a moment longer to realize that the embedding of G_0 is determined as well– this embedding is determined by the cyclic order of (surviving) edges around their vertices. We will have deleted some edges from many of these vertices by the time we reach G_0, potentially leaving many empty connections. However, the cyclic order of the surviving edges won't be affected by the order in which these connections were emptied.

One might worry that the choice of *which*[2] copy of an edge of multiplicity two to delete could affect the embedded isomorphism type after an E_2 or E_3 reduction, but it's easy to check that the two possible reduced configurations are (embedded) graph isomorphic by looking at the pictures above. Formally, the point is that the two copies of the edge are adjacent in the cyclic ordering of edges at each vertex, so the surviving copy is always in the same cyclic position relative to surviving edges incident to the vertex. □

Appendix C: Branch and Bound Algorithm for Expansions

Proposition 17 suggests a strategy for generating diagrams: start by enumerating embedded planar simple graphs of vertex degree ≤ 4 using *plantri* , then expand them to 4-regular embedded planar graphs using the moves above. We can generate embedded isomorphic graphs with different expansion sequences, so we will have to sort the graphs into isomorphism classes. We start with an embedded planar graph of vertex degree ≤ 4.

------

[2] Remember that the choice of "base edge" was arbitrary.

Lemma 18. *If G_0 is obtained from a 4-regular embedded planar multigraph G by the reduction process of Proposition 17 then either every vertex of degree one in G_0 has exactly one loop edge in G and one multiedge of multiplicity two obtained by E_2 or E_3 or the graph is ⊙.*

Proof. If we expand G_0 to G using the four moves, three empty connections on the vertex must be filled during the process. If they are filled by redoubling the existing edge, then the degree of the vertex at the other end of the edge was also one, and we get ⊙. Otherwise, we must fill two by adding a loop edge, and the other by doubling the existing edge. □

We also observe that two pairs of vertices ab and cd on the unit circle may be joined by nonintersecting chords inside the circle if and only if the pairs are unlinked on the circle (⊙ instead of ⊗). This happens when the order of the vertices around the circle is $abcd$ instead of $acbd$ or $adbc$.

We can now design an algorithm to produce all possible expansions of G_0, a given connected embedded planar simple graph of vertex degree ≤ 4 as an integer constraint satisfaction problem. By Lemma 18, we must add a loop to each vertex of degree one in G_0 eventually. We can save time by doing so at the start of the computation. We will therefore assume that loops have been added to create a *prepared* graph G_1, and each vertex has degree 2, 3, or 4.

We will now define four classes of variables:

- l_i for every vertex v_i of degree 2

- $d_{i,j}$ for every non-cut edge e_{ij} in the graph joining vertices of degree < 4.

- $c_{i,j}$ for every cut edge e_{ij} joining vertices of degree < 4.

- $p_{i,j}$ for every pair of vertices v_i, v_j which both have degree 2 and are both on two different faces of the embedding

We take the subscripts to be unordered. That is, $d_{4,17}$ and $d_{17,4}$ are the same variable, since the edges $e_{4,17}$ and $e_{17,4}$ are the same edge.

These variables will all take the values 1 or 0, which represent the presence or absence of E_1 loop edges, E_2 or E_3 doubles of existing edges, and E_4 insertions of new pairs of edges. We can now define two sets of equations relating these variables.

Definition 19. We define the *vertex degree equations* for a prepared graph G_1 to be the collection of equations indexed by the vertices of G_1 given below. For each vertex index i of degree $\delta(i)$

$$\delta(i) + 2l_i + \sum_j d_{i,j} + \sum_j c_{i,j} + 2\sum_j p_{i,j} = 4$$

where the sums are taken over all j for which the appropriate variables exist. These equations express the fact that in a complete expansion, the vertex degrees must all be four.
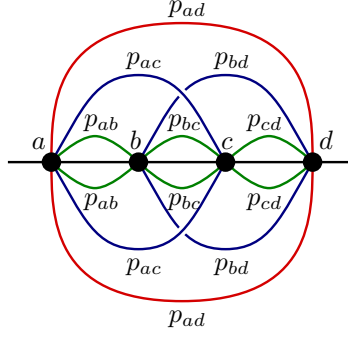
FIG. 21: Suppose that the four vertices a, b, c and d are on a pair of different faces of the graph. There are six potential edges connecting these vertices. The variables p_{ab}, p_{bc}, p_{cd} and p_{da} obey only vertex degree equations. But the variables p_{ac} and p_{bd} obey both vertex degree equations and the additional pair linking equation $p_{ac} + p_{bd} \leq 1$.

The pair variables $p_{i,j}$ satisfy an additional set of equations:

Definition 20. For each $p_{i,j}$ and $p_{k,l}$ so that the vertices v_i, v_j, v_k and v_l are on the same pairs of faces, and so that the vertices are in the (cyclic) order v_i, v_k, v_j, v_l or v_i, v_l, v_j, v_k we have an additional *linking equation*

$$p_{i,j} + p_{k,l} \leq 1$$

These equations express the fact that the edges corresponding to a linked pair of endpoints along a face must intersect inside the face. Therefore, if two pair variables are linked, at most one of them can take the value 1. For instance, in the situation shown in Figure 21 there are four vertices of degree two along a pair of faces, we have six pair variables, two of which obey an additional linking equation.

Proposition 21. *Every assignment of $\{0, 1\}$ to the variables l_i, $d_{i,j}$, $c_{i,j}$, and $p_{i,j}$ which obeys the vertex degree equations and linking equations corresponds to an expansion of the connected planar graph G_1 (with vertex degrees 2, 3, and 4 and loop edges only) to a collection of embeddings for the connected planar 4-regular multigraph G.*

Proof. Actually, there is only a little to check. By the arguments in the proof of Proposition 17, the order of expansion moves is irrelevant. So suppose there are n moves, and fix an order for them. We must show that we can generate a family of graphs $G_1, G_2, \ldots, G_n = G$. If we can perform the indicated expansions at all, we will generate a unique connected 4-regular planar multigraph G (we will see that the embedding of G depends on choices we make along the way). So suppose we have generated a given (embedded) G_i, and are trying to expand to G_{i+1}.

If the next expansion is an E_1 expansion indicated by a positive l_i, it is possible as long as the vertex degree at v_i is small enough. This is true because the corresponding vertex degree equation is satisfied by hypothesis. We must choose which side of the edge to insert the loop; each choice yields a different embedding of G_{i+1}, and following the various possibilities will lead to a family of embeddings for $G_n = G$.

If the next expansion is an E_2 indicated by a positive $d_{i,j}$, it is possible as long as the vertex degrees of v_i and v_j are small enough. This is true by their vertex degree equations. There is only one way to make this expansion, leading to a unique embedding for G_{i+1}.

If the next expansion is an E_2 or E_3 expansion indicated by a positive $c_{i,j}$ variable, it is (again) possible if the vertex degrees at v_i and v_j are small enough (which is again true by the vertex degree equations) *and* if $e_{i,j}$ is a cut edge of G_i. We never apply these expansions more than once to an edge, so $e_{i,j}$ is a cut edge of G_i since it was a cut edge of G_1. Choosing between E_2 and E_3 expansions will yield different embeddings of G_{i+1} and we must follow both possibilities to generate the final family of embeddings of G.

This much was easy. If the next expansion is of type E_4, there is more to check. First, we note that there is no ambiguity in embeddings here: if we can do the E_4 expansion, we can do it in only one way and we generate a unique embedding of the graph G_{i+1}. But can we do it at all? Each E_4 indicated by a positive $p_{i,j}$ requires several conditions. First, vertex degrees at v_i, v_j must be small enough, which is true as usual because the vertex degree equations are satisfied.

We last have only to observe that by the vertex degree equations, the final graph G is a 4-regular planar multigraph. Since we have only added edges along the way, G is connected because G_1 was. $\qquad\square$

We have reduced the problem to that of building and satisfying the vertex degree and linking equations. This problem is basically standard, and we use the usual branch-and-bound algorithm. We must define a canonical order on the variables (it doesn't matter how, but to be specific, in our implementation we sort the classes of variables in the order $l_i \prec d_{i,j} \prec c_{i,j} \prec p_{ij}$ and in dictionary order by the (sorted) pair $\{i, j\}$ within each class). Then we enumerate the possible assignments of $\{0, 1\}$ to variables recursively, pruning the tree whenever a vertex degree or linking equation is violated. As usual, this is in theory possibly exponentially slow, but in practice efficient enough for small n.

We now consider the problem of dividing the results into embedded isomorphism classes. We first observe that we have already shown in Proposition 17 two different reduced graphs G_0 and G_0' cannot expand to the same G since the embedded isomorphism type of the reduction G_0 is determined by the embedded isomorphism type of the expansion. However, it is possible for two different collections of expansion moves for the *same* graph G_0 to produce isomorphic G and G' as in Figure 15 above. Therefore, we insert the various expansions of each graph into a pdstor (see Definition 9) with isomorphism checking and then combine the contents of these pdstors into a

final list of shadows.

---

[1] The On-Line Encyclopedia of Integer Sequences.

[2] F Aicardi. Tree-like curves. In *Singularities and bifurcations*, pages 1–31. Amer. Math. Soc., Providence, RI, 1994.

[3] V I Arnold. Plane curves, their invariants, perestroikas and classifications. *Singularities and Bifwrcations*, 1994.

[4] J Bouttier, P Di Francesco, and E Guitter. Census of planar maps: from the one-matrix model solution to a combinatorial proof. *Nuclear Physics. B*, 645(3):477–499, 2002.

[5] G Brinkmann, G Brinkmann, B D McKay, and B D McKay. Fast generation of planar graphs. *MATCH Commun. Math. Comput. Chem*, 58(2):323–357, 2007.

[6] G Brinkmann, S Greenberg, C Greenhill, B D McKay, R Thomas, and P Wollan. Generation of simple quadrangulations of the sphere. *Discrete mathematics*, 305(1):33–54, 2005.

[7] Harrison Chapman. Asymptotic laws for knot diagrams. in preparation, 2015.

[8] Moshe Cohen and Sunder Ram Krishnan. Random knots using Chebyshev billiard table diagrams. *arXiv.org*, May 2015.

[9] Robert Coquereaux and Jean-Bernard Zuber. Maps, immersions and permutations. *arXiv.org*, July 2015.

[10] Isabel K Darcy. Biological Distances on Knots and Links: Applications to XER recombination. *Journal of Knot Theory and its Ramifications*, 10(02):269–294, March 2001.

[11] Y Diao, C Ernst, and U Ziegler. *Generating large random link diagrams*. Physical and Numerical Models in Knot Theory . . . , 2005.

[12] Yuanan Diao, John C Nardo, Yanqing Sun, and Yanqing Sun. Global knotting in equilateral random polygons. *Journal of Knot Theory and its Ramifications*, 10(04):597–607, 2001.

[13] Nathan Dunfield, A Hirani, M Obeidin, A Ehrenberg, S Bhattacharyya, and D Lei. Random Knots: A preliminary report, 2014.

[14] S F Edwards. Statistical mechanics with topological constraints: I. *Proceedings of the Physical Society*, 91(3):513–519, July 1967.

[15] Chaim Even-Zohar, Joel Hass, Nati Linial, and Tahl Nowik. Invariants of Random Knots and Links. *arXiv.org*, November 2014.

[16] Bruce Ewing and Kenneth C Millett. Computational algorithms and the complexity of link polynomials. Paris, 1997.

[17] S M Gusein-Zade. On the enumeration of curves from infinity to infinity. In *Singularities and bifurcations*, pages 189–198. Amer. Math. Soc., Providence, RI, 1994.

[18] S M Gusein-Zade and F S Duzhin. On the number of topological types of plane curves. *Uspekhi Mat. Nauk*, 53(3(321)):197–198, 1998.

[19] Yoko Hashizume. On the uniqueness of the decomposition of a link. *Osaka Math. J.*, 10:283–300, 1958.

[20] J L Jacobsen and P Zinn-Justin. A transfer matrix approach to the enumeration of knots. *Journal of Knot Theory and its Ramifications*, 11(5):739–758, 2002.

[21] Richárd Kápolnai, Gábor Domokos, and Tímea Szabó. Generating spherical multiquadrangulations by restricted vertex splittings and the reducibility of equilibrium classes. *Periodica Polytechnica Electrical Engineering*, 56(1):11–10, 2012.

[22] Louis H Kauffman. New invariants in the theory of knots. *American Mathematical Monthly*, 95(3), March 1988.

[23] Jenö Lehel. Generating all 4-regular planar graphs from the graph of the octahedron. *Journal of Graph Theory*, 5(4):423–426, 1981.

[24] Xiao-Song Lin and Zhenghan Wang. Integral geometry of plane curves and knot invariants. *arXiv.org*, November 1994.

[25] Matt Mastin. Links and planar diagram codes. *Journal of Knot Theory and its Ramifications*, 24(3):1550016–1550018, 2015.

[26] Brendan D McKay. Isomorph-free exhaustive generation. *Journal of Algorithms*, 26(2), February 1998.

[27] Brendan D McKay and Adolfo Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, 2014.

[28] Moon, Hyeyoung. *Calculating knot distances and solving tangle equations involving Montesinos links*. PhD thesis, University of Iowa, December 2010.

[29] S K Nechaev, A Yu Grosberg, and A M Vershik. Random walks on braid groups: Brownian bridges, complexity and statistics. *Journal of Physics A: Mathematical and General*, 29(10):2411–2433, 1996.

[30] Enzo Orlandini and Stuart G Whittington. Statistical topology of closed curves: Some applications in polymer physics. *Reviews of Modern Physics*, 79(2):611–642, 2007.

[31] Nicholas Pippenger. Knots in random walks. *Discrete Applied Mathematics. The Journal of Combinatorial Algorithms, Informatics and Computational Sciences*, 25(3):273–278, 1989.

[32] M Polyak. Invariants of curves and fronts via Gauss diagrams. *Topology*, 1998.

[33] Gilles Schaeffer. Bijective census and random generation of Eulerian planar maps with prescribed vertex degrees. *Electronic Journal of Combinatorics*, 4(1):Research Paper 20–14 pp. (electronic), 1997.

[34] Gilles Schaeffer and Paul Zinn-Justin. On the asymptotic number of plane curves and alternating knots. *Experiment. Math.*, 13(4):483–493, 2004.

[35] De Witt Sumners and Stuart G Whittington. Knots in self-avoiding walks. *Journal of Physics A: Mathematical and General*, 21(7):1689–1694, 1988.

[36] T.R.S Walsh and A.B Lehman. Counting rooted maps by genus. i. *Journal of Combinatorial Theory, Series B*, 13(3):192 – 218, 1972.