Generating Large Random Knot Diagrams

Yuanan Diao, Claus Ernst, and Uta Ziegler

Abstract. In this paper, we study the problem of generating large random knot diagrams, namely 4-regular planar graphs that are regular projections of prime knots or links. We propose two methods and explore various questions that arise along the way.

1. Introduction

It is well known in knot theory that tabulating knots and links is a very difficult task. Currently, the existing knot tabulation tables are all based on the crossing numbers of the knots. The number of knots and links with crossing number n increases exponentially and it quickly becomes impractical to tabulate all knots and links of the given crossing number. Consequently, not much is known about large knots. Let $\mathcal{L}_n$ be the set of all links with crossing number n and let $\mathcal{AL}_n$ be the set of all alternating links, then most members of $\mathcal{L}_n$ are non-alternating composite links. Let $\mathcal{K}_n$ ($\mathcal{AK}_n$) be the set of all (alternating) prime knots with crossing number n. A typical member of $\mathcal{K}_n$ is a non-alternating prime knot, that is, probably hyperbolic. If n is large, then the size of the set $\mathcal{K}_n$ is very large and the size of $|\mathcal{AL}_n|$ can be approximated as follows [**ST**]:

$$\frac{a_{n-1}}{8(2n-3)} \leq |\mathcal{AL}_n| \leq \frac{a_{n-1}}{2},$$

where $a_n \approx \frac{11.5}{4\sqrt{\pi}} n^{-5/2} 6.148^{n-3/2}$. Furthermore it is known that the ratios of $\frac{|\mathcal{AK}_n|}{|\mathcal{K}_n|}$ and $\frac{|\mathcal{AL}_n|}{|\mathcal{L}_n|}$ approach zero as n grows to infinity [**T**].

While there are many different measures of knot complexity (such as the genus, the unknotting number, the bridge number, the ropelength, the knot energies), the crossing number of the knot (link) is probably the most often used. In this paper, when we speak of a large knot or link, we mean a knot or a link with a large crossing number. It is a fundamental problem in knot theory to find the relations between the crossing number and the other knot complexity measures. The authors are particularly interested in the relations between the crossing number and the ropelength of a knot and this is a main motivation of this paper. While this is a

1991 *Mathematics Subject Classification.* Primary 57M25.

Key words and phrases. Keywords: Knot diagrams.

The authors are partially supported by NSF grant DMS-0310562.

very hard question in general, much has been achieved despite the difficulties. It is shown in [**D**] that the ropelength $L(K)$ of a knot K is bounded below by

$$L(K) \geq \frac{1}{2}\left(17.334 + \sqrt{17.334^2 + 64\pi Cr(K)}\right). \tag{1.1}$$

For relatively small knots, the above formula produces the best known theoretical lower bounds of the ropelengths of these knots. The lower bounds of the ropelengths of knots with crossing number up to 20 produced by this formula are listed below.

$Cr(K)$	=	3	4	5	6	7	8	9	10	11
$L(K)$	>	23.70	25.29	26.74	28.08	29.33	30.51	31.64	32.70	33.73

$Cr(K)$	=	12	13	14	15	16	17	18	19	20
$L(K)$	>	34.71	35.66	36.58	37.46	38.32	39.16	39.97	40.76	41.54

For large knots, the above lower bound is inferior to the following bound obtained by Buck and Simon [**BS**]:

$$L(K) \geq b\,(Cr(K))^{\frac{3}{4}}, \tag{1.2}$$

where b is some positive constant. The constant b is estimated to be at least 1.105 by a result in [**BS**] and it is improved to 2.135 in [**RS**]. This three fourths power is also shown to be achievable for some knot families ([**CKS**], [**DE**]). That is, there exists an infinite family $\{K_n\}$ of knots and a constant $a_0 > 0$ such that $Cr(K_n) \to \infty$ as $n \to \infty$ and $L(K_n) \leq a_0 \cdot (Cr(K_n))^{3/4}$. On the other hand, it is also shown in [**DEY**] that $L(K)$ is bounded above by $c(Cr(K))^{3/2}$ (where $c > 0$ is a constant). It is widely suspected that this 3/2 power can be reduced and it is possible that the ropelength of any knot K may be bounded above by $O(Cr(K))$ or $O(Cr(K)\ln^p Cr(K))$ for some positive power p. It is shown recently that the power 1 can be realized by a family of knots [**DET**] and any power between 3/4 and 1 can be realized by some family of knots. However, what happens between the powers 1 and 3/2 remains a challenging problem.

In general, relating a different measure to the crossing number is difficult since in order to obtain information on how these measures of knot complexity grow with the crossing number n, one is presented with a host of seemingly intractable problems. Foremost, determining the dependent variable n proves to be impossible in practice. On the one hand, deciding the minimum crossing number n for a given large knot K cannot be done easily. So when there are many such knots to be analyzed, the required computing time would exceed our limit quickly. On the other hand, it is also a difficult task to create an arbitrary large knot with a given crossing number. It is relatively easy to construct certain knots with the given crossing number, such as an alternating knot. But that contradicts our intention to select an arbitrary knot.

It is natural for one to turn to numerical methods in an effort to gain certain insight into the issue. However, in the case of relating crossing number and other knot complexity measures, this approach has not gained much momentum. The chief reason is the lack of an effective way of generating large random knots.

With the long-term goal of answering how the measures of complexity grow for members of $\mathcal{A}_n$ with n, this paper addresses the short-term goal of generating

members of the set $\mathcal{A}_n$ with a computer. More specifically, we propose two methods that generate regular projections of knots. These regular projections can be viewed as 4-regular planar graphs G with n vertices. Such graphs are called regular projection graphs or RP-graphs for short. An RP-graph G can be transformed easily into a knot or link L by simply adding the overpass/underpass information to each vertex. Assigning the overpass/underpass information at each vertex of G in a random way is likely to result in a knot or link L whose crossing number is smaller than the number of crossings in the diagram. However, if the overpass/underpass assignment to each vertex of G can be made in an alternating way and the RP-graph G is 4 edge-connected, then the obtained alternating knot is a prime knot and thus a member of $\mathcal{AK}_n$ and the diagram would represent a minimum projection of certain knot. In this paper, we are mainly concerned with the problem of how to generate the random knot diagrams, not how to realize a knot with the given crossing number from such a projection diagram.

The first method to generate an RP-graph presented here is based on the principle of a *blossom tree* [**ST**]. With a runtime of $O(n)$ a 4-regular planar graph with n vertices is randomly generated. If this graph is the regular projection of a link with k components then the components can be determined in $O(n)$ steps and in $O(n^3)$ time this graph can be modified into a graph that is the projection of a knot. The final product of the method is an RP graph that is a regular projection of an alternating prime knot. The number of vertices in this RP-graph is less than or equal to n.

A main result in [**DEY**] is that for any knot K with crossing number n, there exists a knot K' that is topologically equivalent to K such that K' has a regular projection that is Hamiltonian and at most $4n$ crossings in this regular projection. The algorithm used in [**DEY**] to produce the $n^{3/2}$ power upper bound for $L(K)$ is based on the existence of Hamilton cycle in an RP-graph of K'. The problem of finding a Hamilton cycle in a planar graph is known to be NP-complete [**K**]. Even for the more restricted class of 4-regular planar graphs finding a Hamilton cycle is still a difficult problem. Because of this, the second method introduced here may have a special advantage since it generates 4-regular planar graphs G based on an existing Hamilton cycle in it: the Hamilton cycle could be used in further computations (such as computing an upper bound on the ropelength). Although this method has a larger average run time than the first method, it may be well worth the effort.

In the next section, we will introduce some basic concepts. We will then describe the two methods in Section 3 and Section 4. Section 5 contains some key numerical results comparing the two methods.

2. Basic Concepts

A *graph* G consists of a set $V(G)$, a set $E(G)$, and an incidence relation which says that every element of $E(G)$ is incident with two elements of $V(G)$. The elements of $V(G)$ are called the *vertices* of G, and $V(G)$ is called the *vertex set* of G. The elements of $E(G)$ are called the *edges* of G, and $E(G)$ is called the *edge set* of G.

Let G be a graph. The *degree* of a vertex v of G is the number of edges of G incident with v. The graph G is called *k-regular* if every vertex of G is of degree k. Let e be an edge of G incident with vertices u and v. We say that e *connects* u and v, u and v are the *ends* of e, and u and v are *adjacent*. We also use uv or vu to denote e when there is only one edge connecting u and v. If $u = v$ then e is called a *loop edge*.

A graph H is a *subgraph* of a graph G if $V(H) \subset V(G)$ and $E(H) \subset E(G)$. In the case that $V(H) = V(G)$, H is called a *spanning subgraph* of G. Two graphs G and H are said to be *isomorphic* if there exist bijections $f: V(G) \longrightarrow V(H)$ and $g: E(G) \longrightarrow E(H)$ such that $e \in E(G)$ is incident with x, $y \in V(G)$ if, and only if, $g(e) \in E(H)$ is incident with $f(x)$, $f(y) \in V(H)$.

A *path* P of length $k-1$, where $k \geq 2$, is a graph which is isomorphic to the graph with vertex set $\{v_1, v_2, ..., v_k\}$ and edge set $\{v_i v_{i+1} : i = 1, \ldots, k-1\}$. We say that v_1 and v_k are the *ends* of P, P is *from* v_1 *to* v_k, and P is *between* v_1 *and* v_k. A *cycle* C of length k, where $k \geq 3$, is a graph which is isomorphic to the graph with vertex set $\{v_1, v_2, ..., v_k\}$ and edge set $\{v_i v_{i+1} : i = 1, \ldots, k-1\} \cup \{v_k v_1\}$. A *Hamilton cycle* in a graph G is a spanning subgraph that is a cycle. A graph with a Hamilton cycle is said to be *Hamiltonian*. A graph with no cycle is a *tree*. A tree is *rooted* if there is a single vertex in the tree labelled as root.

A *geometric realization* of a graph G in R^2 or R^3 is such that the vertices of G are represented by distinct points in the space, every edge of G is represented by a simple curve in R^2 or R^3 connecting the two points representing its ends, and these simple curves representing the edges do not intersect each other except possibly at their ends. We say that a graph G is *planar* if it has a geometric realization in a plane. Such a geometric realization is called a *plane graph*. A planar *map* consists of a planar graph and a cyclic order of the edges around each vertex.

4-regular plane graphs are related to knots and links as follows: A projection p of a knot or link L in R^3 is a continuous function $p : R^3 \to R^2$. The image $p(K)$ is called the projection of the knot or link K into a plane. $p(K)$ is a closed curve (or collection of closed curves if K is a link) in R^2 that may contain self-intersecting points. A self-intersecting point is also called a *crossing* of the projection. The *multiplicity* of a crossing in the projection is the number of strands that pass through that point. We say that a projection is a *regular projection* if there are only finitely many crossings in $p(K)$ and all crossings are of multiplicity 2, that is there are exactly two points on L that map to any of the crossings. It is a well-known result in knot theory [**BZ**] that for any knot in R^3 most projection directions give rise to regular projections. Furthermore, a regular projection $p(K)$ of K is called a *minimum projection* of K if it is a regular projection with $Cr(K)$ crossings.

Let K be a knot or link and let G be a regular projection of K. If we treat the crossings in G as vertices and the arcs of G joining these crossings as edges, then G can be viewed as a 4-regular plane graph. Thus, from now on, we may view a regular projection G as a 4-regular plane graph G called an *RP-graph* of K, where "RP" serves as a reminder that the graph is obtained as a regular projection of K. If G arises from a minimum projection of K, we then call it a *minimum RP-graph* of K. Note that any 4-regular plane graph is an RP-graph of some knot or link. A 4-regular planar RP-graph is equivalent to a 4-regular planar map in the context of this paper and we use the terms interchangeably.

For any $X \subset V(G)$, let $G - X$ denote the subgraph of G obtained from G by deleting vertices of G in X and edges of G with at least one end in X. Similarly, for any $Y \subset E(G)$, we use $G - Y$ to denote the subgraph of G obtained from G by deleting the edges in Y (but keeping all vertices of G). We say that G is *k-connected*, where k is a positive integer, if $|V(G)| \geq k + 1$ and, for any subset $X \subset V(G)$ with $|X| < k$, $G - X$ is connected. We say that G is *k-edge-connected* if, for any $Y \subset E(G)$ with $|Y| < k$, $G - Y$ is connected. The *connectivity* (respectively, *edge-connectivity*) of G is the largest integer k such that G is k-connected. It is easy to see that a 4-regular plane graph is either 2-edge connected or 4-edge connected. If G is a 2-edge connected 4-regular plane graph that arises as a regular projection from some knot or link K, then either G is not a minimum RP-graph of K or K is not a prime knot or link. This motivates the following definition: A 4-regular plane graph is *diagrammatically prime* if it is 4-edge connected.

Let G be a 4-regular RP-graph, let v be a vertex of G, and let e_1, e_2, e_3, e_4 be the edges of G incident with v. Suppose that e_1, e_2, e_3, and e_4 occur around v in this cyclic order as shown in Figure 1. Then we say that e_i is *opposite* to e_j if $|j - i| = 2$, and e_i and e_j are *adjacent* otherwise.

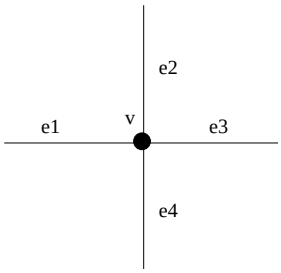

FIGURE 1. Pairs of opposite edges: $\{e_1, e_3\}$, $\{e_2, e_4\}$.

One can travel along the edges of a 4-regular plane graph G if one follows the convention that if one travels an edge e leading to a vertex v then one has to continue along the edge opposite to e. In this way G can be viewed as the union of several closed curves. We say that G has k *components* if G is the union of k closed curves. We say that G is a *knot graph* if the number of components in G is one. Of course, the number of components of G is the same as the number of components in any link K which has G as an RP-graph.

In the following two sections we give two methods that can be used to computer generate large diagrammatically prime 4-regular plane knot graphs G in a random manner. However in the generation process the number of vertices in the graphs varies from one graph to the next. If the user wants a specific interval into which the number of vertices of a generated graph should fall then one needs to run the process several times until a graph with the desired number of vertices is generated.

3. An algorithm to generate large, diagrammatically prime knot graphs

This method first generates a 4-regular planar map with n vertices (Step 1), combines the components of this RP-graph into a knot graph (Step 2) and decomposes the knot graph into diagrammatically prime knot graphs (Step 3).

Step 1. Creating a 4-regular planar map. Here we deploy an algorithm introduced in [**SZ, S**] to generate 4-regular planar RP-graphs with n vertices in $O(n)$ time. A short outline of the algorithm is given below, for the details please refer to [**SZ, S**]. In the following definition, two labels are used to mark the vertices of a graph: one is called a *bud* and the other is called a *leaf*. In Figure 2 below, a bud is marked by an arrow and a leaf is marked as a solid dot. A vertex with degree more than 1 is marked by a circle.

DEFINITION 3.1. A *blossom tree* is a rooted plane tree such that:
(i) vertices of degree one are marked as either buds or leaves;
(ii) every inner vertex has degree four and is incident to exactly one bud;
(iii) the root is a leaf.

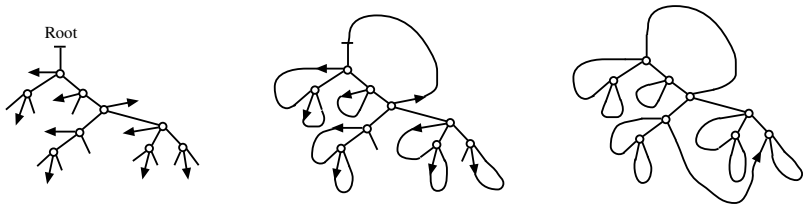

FIGURE 2. On the left is a blossom tree with 9 vertices, 11 leafs and 9 buds. One leaf is labelled as the root. The figure in the center shows how the blossom tree can be changed into a 4 regular plane graph by connecting each bud to the nearest leaf in counterclockwise direction. This operations leaves two leafs remaining, which are connected by an oriented edge. On the right the final rooted 4 regular map is shown.

To generate a blossom tree, we first generate a random binary plane tree (a tree such that all internal vertices have degree three) with n internal vertices and $n+2$ leaves (one of which is the root of the tree). Now at each internal vertex one can add an edge whose end is a bud, randomly in one of three different ways, see Figure 2. It has been shown that there are exactly $\frac{3^n}{n+1}\binom{2n}{n}$ rooted blossom trees with n internal vertices. Furthermore, each of which can be generated with equal probability. The *closure* of a blossom tree is created by connecting the n buds with the nearest unmatched leaf in a counterclockwise direction. (This procedure uses a counterclockwise order of the leaves and buds of the tree around the infinite face the tree defines in the plane.) After all buds have been matched to leaves, there are two unmatched leaves in the infinite face. Finally after closing the two unmatched leaves with an edge one obtains a 4-regular rooted planar map. **Rooted means that one edge has an orientation.??? Shouldn't this read like: The edge that contains the root leaf has a naturally defined orientation.? Please correct.** The number of such maps is $\frac{2}{n+2}\frac{3^n}{n+1}\binom{2n}{n}$ and such a map can

be constructed in $O(n)$ time. Moreover the algorithm generates such a map with a uniform probability distribution on the space of rooted 4-regular planar maps with n vertices. In our case, the arrows (which denote the buds) in the maps are deleted since we are not interested in rooted 4-regular planar maps but just in 4-regular planar graphs. Deleting the orientation from this single edge results in a non-uniform distribution on the space of 4-regular planar graphs. Thus the 4-regular planar RP-graphs are generated with a non-uniform distribution.

Step 2. Creating a knot graph: The 4-regular RP-graphs so generated are usually not diagrammatically prime and not a knot projection graph either since it usually has many components. Numerical experiments (see [**SZ**]) show that a 4-regular planar map with 2^{10} vertices generated this way has an average number of about 51 components, and a 4-regular planar map with 2^{12} vertices so generated has an average of about 199 components. In fact, it is conjectured that the number of components in such a 4-regular planar map is proportional to the number of vertices of the graph [**SZ**]. Since we are interested in diagrammatically prime RP-graphs of knots, we need to modify the RP-graphs into RP knot graphs.

Given a 4-regular planar map G with n vertices and k components. The idea is to combine components in a manner which does not modify the number of vertices while ensuring that the modified RP-graph remains a 4-regular planar map. This can be done if a face of G is bound by edges of at least two different components of G. Walking around a face of G in one direction, the edges v_1v_2 and w_1w_2 are found, where, for both edges, the subscript 1 indicates the vertex first encountered. If the two edges are disjoint then we delete the two edges and insert two new edges v_1w_2 and w_1v_2 as shown in the left half of Figure 3. This edge replacement reduces the number of components by one, but does not reduce the number of vertices. If no such disjoint edges can be found on any face of G and there is still more than one component in G, then there are two edges that intersect at either one or both of their endpoints. In this case an edge replacement move is still possible and results in the modifications shown in the right half of Figure 3. However such a move introduces one or two loop edges. Continuing this process one obtains a 4-regular graph with n vertices and one single component.

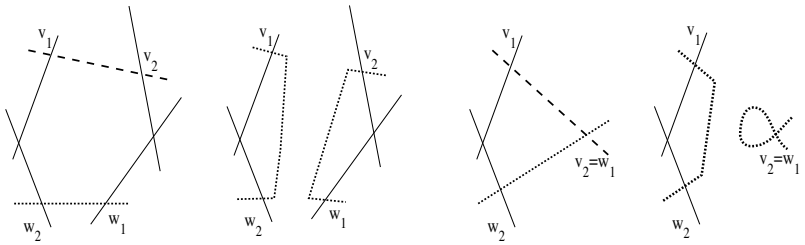

FIGURE 3. An edge replacement move. On the left using two disjoint edges v_1v_2 and w_1w_2, on the right two edges v_1v_2 and w_1w_2 which have one vertex in common. In this case a replacement move introduces one loop edge.

To combine the components randomly, a vertex v in G is picked at random where two different components of G cross at v. One of the faces of G containing v

is picked at random. We now search the edges of this face. If this face contains two disjoint edges of different components of G a replacement move of the above type is carried out. If the face does not contain two disjoint edges of different components of G one of the other faces of G containing v is chosen. If none of the faces of G containing v can be used for a replacement move then a vertex different from v is picked at random and the procedure is repeated. If during this process two edges along a common face of different components of G are detected which are not disjoint, then this is stored and the search for a replacement move continues. A replacement move leading to a loop edges as shown on the right in Figure 3 is only carried out if no replacement move using two disjoint edges is available. This process of combining the components can be carried out in $O(n^3)$ steps.

The exact number of 4-regular planar maps G of n vertices with one component is unknown, and no claim is made that the above algorithm produces such graphs with a uniform distribution for a given vertex number n. Clearly any 4-regular planar one-component map with n vertices and without loop edges can be obtained by the algorithm (with certain probability). Some graphs containing loop edges may also be produced, however. Therefore, we cannot claim that this procedure will always generate a 4-regular planar one-component map with n vertices and without loop edges. Notice that in the algorithm, a loop edge is avoided whenever possible. This is no drawback since any minimal RP-graph of a knot K does not contain loop edges and we generally prefer knot RP-graphs whose number of vertices are close to (if not equal to) the crossing number of the corresponding knots.

Step 3. The knot projection graphs obtained up to this point are not necessarily diagrammatically prime. To obtain diagrammatically prime 4-regular graphs a given 4-regular planar knot graph is split into 4-edge connected components using an algorithm given in [**DT**], which can be carried out in $O(n^2)$ steps. In this last step, the control over the number of vertices in the final diagrammatically prime knot graphs is lost in that a graph with a fixed number of n vertices now fall apart into several graphs of varying sizes. See Section 5 for further discussions.

4. An algorithm to generate large diagrammatically prime Hamiltonian knot graphs

In this section, we discuss the second method that generates diagrammatically prime knot graphs G with a genetically built in Hamilton cycle. This method relies on the fact that every knot K has a regular projection of at most $4Cr(K)$ crossings with a Hamilton cycle [**DEY**]. The main idea is to start with a Hamilton cycle of n vertices (Step 1) and add edges to the cycle in a random manner to create a 4-regular graph (Step 2). The resulting graph is modified into a 4-regular planar graph (Step 3), whose components are combined into a knot graph (Step 4) and finally the knot graph is decomposed into diagrammatically prime knot graphs (Step 5). Although it is far from clear how the knots generated in such a manner are distributed (as with any other method at this point), all knots can be generated in such a way, though the generated knot projections are usually not the minimum projections. Figure 4 shows a typical diagram generated with this method.

Step 1. Start with a Hamilton cycle with n vertices v_1, v_2, ..., v_n such that the order of the vertices matches the order of v_1, v_2, ..., v_n on the Hamilton cycle.

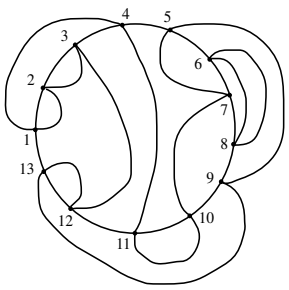

FIGURE 4. A small diagrammatically prime RP-graph of a knot with 13 crossing generated using method 2. The Hamilton cycle is the circle and the vertices are numbered along the cycle.

Step 2. Notice that at each vertex, two edges have to be added. Since each edge added is incident to two vertices, a total of n edges need to be added. These n edges are divided into two groups: those in the region bounded by the Hamilton cycle (called B-edges) and those in the unbounded region (called U-edges). Each such edge e can be represented by a triple of integers of the form (i, j, k). Here, the indices i and j imply that e is incident to vertices v_i and v_j, $k = 0$ if e is a U-edge and $k = 1$ if e is a B-edge. For example, $(2, 5, 0)$ means an outside edge incident to vertices v_2 and v_5 and $(1, 8, 1)$ means the inside edge incident to vertices v_1 and v_8.

The following procedure generates random triples for each of these n edges:

Sub-step 1: Generate a random permutation of the vector $(1, 1, 2, 2, ..., n, n)$. The result is a vector of $2n$ entries: $(j_1, j_2, ..., j_{2n})$. The n edges not on the Hamilton cycle are to be chosen as the pairs (j_1, j_2), (j_3, j_4), ..., (j_{2n-1}, j_{2n}).

Sub-step 2: Generate a random vector $(k_1, k_2, ..., k_n)$ where each k_i is 0 or 1 with equal probability.

Sub-step 3: Merge the two vectors to create the n triples of integers representing the edges which need to be added: (j_1, j_2, k_1), (j_3, j_4, k_2), ...,(j_{2n-1}, j_{2n}, k_n).

Steps 1 and 2 in this algorithm can be carried out using $O(n)$ steps. Up to this point in the algorithm we have a uniform distribution among all possible configurations of the n edges not on the Hamilton cycle. Furthermore, we can always arrange the numbers so that $j_1 \leq j_2$, $j_3 \leq j_4$ and so on, since this does not change the definition of the edge. Notice that at this stage of the algorithm there are several problems. The graph generated so far may have several components. In addition, it may be impossible to embed in R^2 the n edges given by the list (j_1, j_2, k_1), (j_3, j_4, k_2), ...,(j_{2n-1}, j_{2n}, k_n) without many self intersections of these edges. Thus the graph defined by these edges may not be a planar graph. In fact there may be many non equivalent ways to use the triples (j_1, j_2, k_1), (j_3, j_4, k_2), ...,(j_{2n-1}, j_{2n}, k_n) to create nonequivalent (up to isotopy) non-planar 4-regular graphs. In the next step we change the list (j_1, j_2, k_1), (j_3, j_4, k_2), ...,(j_{2n-1}, j_{2n}, k_n) to a new list of n triples whose edges can be embedded into the plane in a unique way without any intersections of these edges (except at the vertices on the Hamilton cycle, of course).

Step 3. It can be easily shown that two edges generated by the triples $e = (i_1, j_1, k_1)$ and $f = (i_2, j_2, k_2)$ must intersect each other in their interior if and only if

a. $k_1 = k_2$ (e and f are both U-edges or both B-edges) and

b. either

$$i_1 < i_2 < j_1 < j_2 \text{ or } i_2 < i_1 < j_2 < j_1. \tag{4.1}$$

Now we randomly pick up an edge e and check if the above conditions are satisfied for any of the other edges. If the conditions are never satisfied for any of the other edges, we are done with e; e remains unchanged. We mark e and randomly select a new edge e that is not already marked. If two triples $e = (i_1, j_1, k)$ and $f = (i_2, j_2, k)$ satisfy (4.1), then with equal probability, we change the two triples e and f into either (i_1, i_2, k), (j_1, j_2, k) or (i_1, j_2, k), (i_2, j_1, k).

LEMMA 4.1. *The above procedure reduces the number of pairs of triples which satisfy the above conditions by at least one.*

PROOF. Assume that $e = (i_1, j_1, k)$ and $f = (i_2, j_2, k)$ satisfy (4.1). Clearly the conflict between e and f is removed in that none of the two potential new pairs of triples (i_1, i_2, k), (j_1, j_2, k) or (i_1, j_2, k), (i_2, j_1, k) satisfies (4.1).

It suffices to show two conditions:

(i) If there is an edge $h = (i_3, j_3, k)$ such that neither h and e nor h and f satisfy (4.1), then none of the 4 potential new triples (i_1, i_2, k), (j_1, j_2, k) or (i_1, j_2, k), (i_2, j_1, k) together with h satisfy (4.1).

(ii) If there is an edge $h = (i_3, j_3, k)$ such that either h and e or h and f satisfy (4.1), then at most one of each pair of the 4 potential new triples (i_1, i_2, k), (j_1, j_2, k) or (i_1, j_2, k), (i_2, j_1, k) together with h satisfy (4.1).

In case (i), there are two subcases $i_1 < i_2 < j_1 < j_2$ and $i_2 < i_1 < j_2 < j_1$. Since they are similar, we will only prove the subcase $i_1 < i_2 < j_1 < j_2$. The situation is shown in Figure 5 (on the left) assuming that e, f and h are B-edges. (The situation is similar if all three are U-edges.) Since h does not satisfy (4.1) when paired with either e or f, we must have that i_3 and j_3 are between either i_1 and i_2 or i_2 and j_1 or j_1 and j_2 or j_2 and i_1 when the vertices are viewed along the Hamilton cycle. Thus h and any of the four potential new triples does not satisfy (4.1).

The proof of case (ii) involves a similar analysis as in case (i) above and is left to the reader. □

Figure 5 (on the right) shows that one such step could reduce the pairs of triples which satisfy the above condition by more than one.

Once we changed two triples satisfying the above condition into two different triples, we start all over again. That is we randomly pick up an unmarked edge e and check if the above conditions are satisfied for any of the other edges. We continue this process until all edges are marked. Now we have a set of n triples which contain no pair satisfying (4.1). Such a set of n triples defines up to isotopy a 4-regular plane graph with n vertices and a Hamilton cycle.

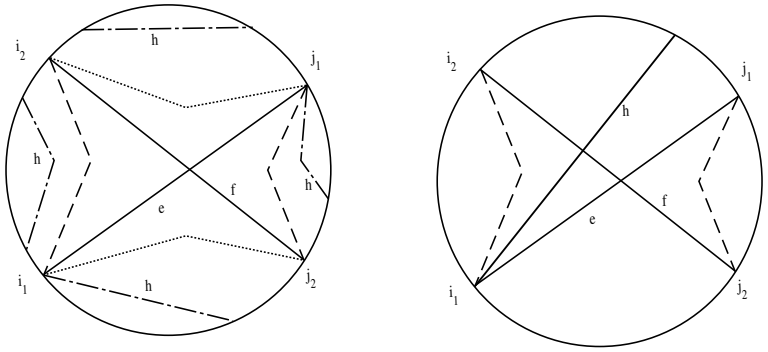

FIGURE 5. On the left: The edges e and f satisfying the condition (4.1) are shown together with the four new triples (i_1, i_2, k), (j_1, j_2, k) (dashed line) or (i_1, j_2, k), (i_2, j_1, k) (dotted line). In addition 4 potential new edges h are shown which do not satisfy (4.1) with any of the other shown edges. On the right: replacing the edges e and f with the two dashed edges also eliminates the intersection between the edges e and h.

Since there are at most $O(n^2)$ pairs of triples satisfying the above condition, this step of the algorithm requires a polynomial time of the order $O(n^3)$ (finding a pair of triples satisfying the above condition takes up to $O(n)$ steps, resulting in an order $O(n^3)$ in the worst case).

Step 4. Creation of a knot graph. The situation now is similar to Step 2 in the first method. A similar procedure as in method one is carried out. The difference is that we cannot use edges that are on the Hamilton cycle in any edge replacement move since this would destroy the Hamilton cycle. Thus on one face of the graph G one needs to find two edges that belong to different components of G and neither edge being on the Hamilton cycle before carrying out an edge replacement move. Otherwise the algorithm is unchanged. In the Lemma below we show that this algorithm almost always works.

LEMMA 4.2. *The algorithm leads to a knot graph G with an Hamilton cycle in all but one case.*

PROOF. If in Step 2 a graph G is created in which the cycle v_1, v_2, ..., v_n forms a component (that is, there is exactly one U and one B edge attached at each v_i), then the algorithm fails. This happens since we only do edge replacement moves with edges that are not on the Hamilton cycle and the Hamilton cycle thus remains one component. In this case the algorithm produces a two-component graph with one component containing the Hamilton cycle and the other containing all the edges which are not on the Hamilton cycle.

We need to show that the algorithm works in all other cases. Notice that if all the U-edges do not belong to one component, then G has a face in the U region that contains two edges not on the Hamilton cycle of different components. This implies that an edge replacement move is possible. So if the algorithm is carried out and there is no further edge replacement left, then all the U-edges must belong to one component. Similarly all B edges must belong to one component of G. We

claim that all edges on the Hamilton cycle belong to one component. If we walk along an edge $v_{i-1}v_i$ then we want to show that the edge v_iv_{i+1} belongs to the same component. If at v_i we have a B-edge and a U-edge then this is obvious. If there are two B-edges (or two U-edges) at v_i then both edges of the Hamilton cycle must be of the same component due to the the fact that both B-edges (or U-edges) belong to the same component.

Now consider the component C of G containing the Hamilton cycle. If there is a single vertex on the Hamilton cycle which is attached to two B edges (or two U edges), then C contains the Hamilton cycle and all the B edges (or all U edges). It is now easy to see that C must indeed be equal to the whole graph G. □

Notice that the special case when one U and one B edge is attached at each vertex v_i happens only if the vector $(k_1, k_2, ..., k_n)$ generated in sub-step 2 of step 2 has an equal number of ones and zeros. Moreover after the merging of the vector $(k_1, k_2, ..., k_n))$ with the pairs (j_1, j_2), (j_3, j_4), ..., (j_{2n-1}, j_{2n}) to the vector (j_1, j_2, k_1), (j_3, j_4, k_2), ..., (j_{2n-1}, j_{2n}, k_n) each vertex v_i shows exactly once in a triple $(j_i, j_{i+1}, 0)$ and once in a triple $(j_k, j_{k+1}, 1)$ for some indices i and k. It can be shown the probability for such an event to happen is exponentially small. For the large number of vertices (see section 5) which we are interested in, the algorithm never encountered this situation in our simulations. If such a rare event does happen, the algorithm simply jumps back to step 1 and generates an entirely new graph.

Step 5. The planar graphs obtained up to this point are not necessarily diagrammatically prime. To obtain diagrammatically prime 4-regular RP-graphs the same algorithm as in Step 3, method one can now be carried out to decompose the graph into parts that are diagrammatically prime. Note that there are some differences to method one however. First in method one it is possible that the graph G obtained at this stage is 1-connected. Here, a graph that is Hamiltonian can still have loop edges but it must be 2-connected. Furthermore the Hamilton cycle naturally breaks apart into smaller Hamilton cycles for each of the diagrammatically prime pieces. Thus the diagrammatically prime graphs obtained are still Hamiltonian.

5. Data and comparisons

Using 10,000 runs for varies initial values of n, large samples of diagrammatically prime RP-graphs are generated. In both methods one cannot control the exact number of crossings in the diagrammatically prime RP-graphs since a reduction procedure is required for the original constructed planar maps with n vertices. Figure 6 shows the frequency of diagrammatically prime RP-graphs generated using method one, while Figure 7 shows the same information using method two. Figure 7 does not contain the graph for $n = 10000$ since the runtime of method two is quite large, see also Figure 11. One would expect that the larger diagrammatically prime RP-graphs are less frequent than smaller diagrammatically prime RP-graphs. However our data as shown in Figure 6 for $n = 1000$, 3000 and 7000 indicates that this ratio (of the larger diagrammatically prime RP-graphs and the total number of graphs generated) does not behavior linearly: there is a pronounced

maximum. This maximum may occur because the vertex numbers of the diagrammatically prime RP-graphs for a given n are dependent on each other. It seems that these maxima become less pronounced as n becomes larger. Most of the diagrammatically prime RP-graphs are quite small. For example in method one the $10,000$ runs for $n = 3000$ generated a total of $1,009,728$ diagrammatically prime RP-graphs; and of those only $27,820$ (or 2.76 percent) have more than 30 vertices. For $n = 500$ about 7.6 percent of the diagrammatically prime RP-graphs generated have more than 30 vertices. This indicates that as n grows, the percentage of large diagrammatically prime RP-graphs declines, while the overall number of large diagrammatically prime RP-graphs increases since many more graphs are generated for larger values of n, see Figure 8.

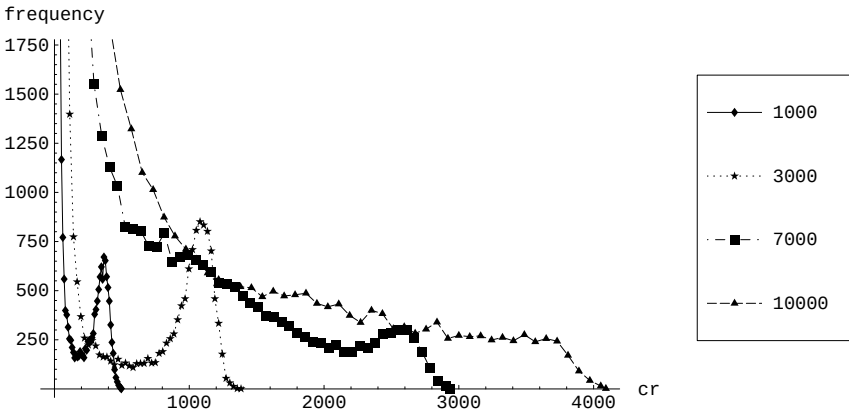

FIGURE 6. The frequency of diagrammatically prime RP-graphs generated using method one and $n = 1000$, 3000, 7000 and $10,000$ for the initial number of vertices. Each curve is based on $10,000$ runs.

The behavior of the Hamiltonian RP-graphs generated using method 2 as shown in 7 is similar as the behavior we have seen for the RP-graphs generated using method one. However there are more diagrammatically prime RP-graphs generated and some of these have a larger vertex number than any of the RP-graphs generated with method one.

Our data as shown in Figure 8 indicates that more diagrammatically prime RP-graphs are generated per run using method two. Moreover for both methods it appears that the number of diagrammatically prime RP-graphs in a given one component RP-graph generated by method one or two (before the last step in the algorithms) grows at the order $O(n)$.

Figure 9 shows that method two generates larger diagrammatically prime RP-graphs on the average. In fact the vertex number of the largest components generated with method two were almost twice the vertex number of the largest component generated using method one. Of course, these numbers are also a function of the number of trials and one would expect that as the number of runs tends to infinity, the size of the largest diagrammatically prime RP-graph would be close to n itself.

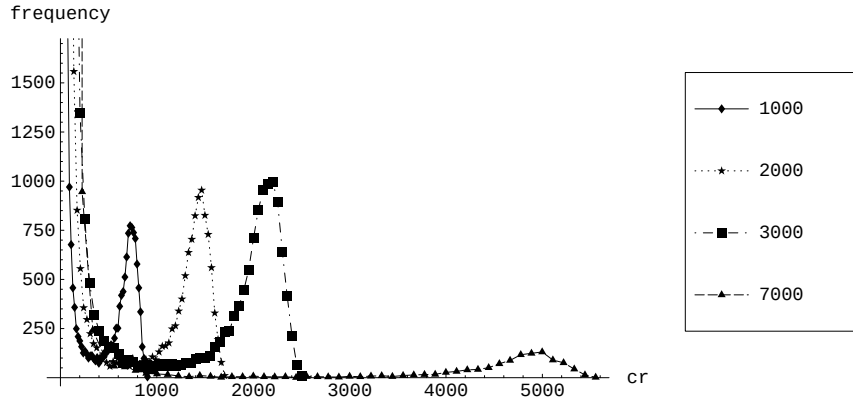

FIGURE 7. The frequency of diagrammatically prime RP-graphs generated using method two and $n = 1000$,2000, 3000 and 7000 for the initial number of vertices. Each curve is based on $10,000$ runs.

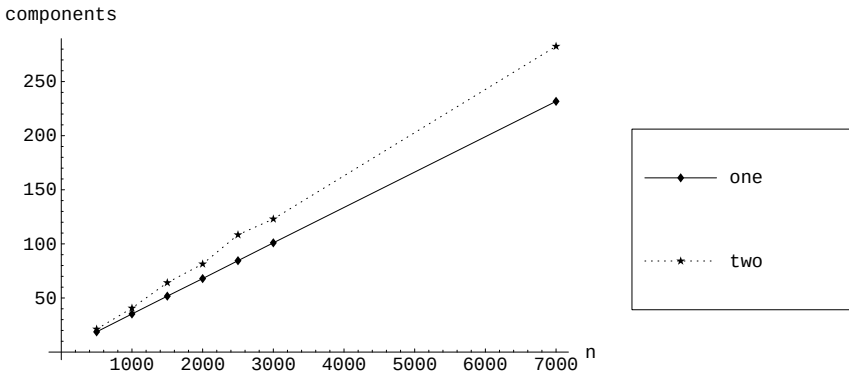

FIGURE 8. The average number of diagrammatically prime RP-graphs generated with one attempted for both methods and different values of n. Each data point is based on $10,000$ runs.

Figure 10 shows the average size of a diagrammatically prime RP-graphs generated using both methods. It appears that either this average size grows extremely slowly or for any large n the average size is bounded above by a different constant for each method. The average size for $n = 7000$ is 16.15 for method one and 24.77 for method two.

Figure 11 shows a comparison of the average run-time of both methods. One can see that method two has a much longer average run-time. Elementary data fitting suggests that the runtime of method one grows as $3.6\times10^{-8}n^2$, while method two has a run-time that grows as $6.8 \times 10^{-9}n^3$.

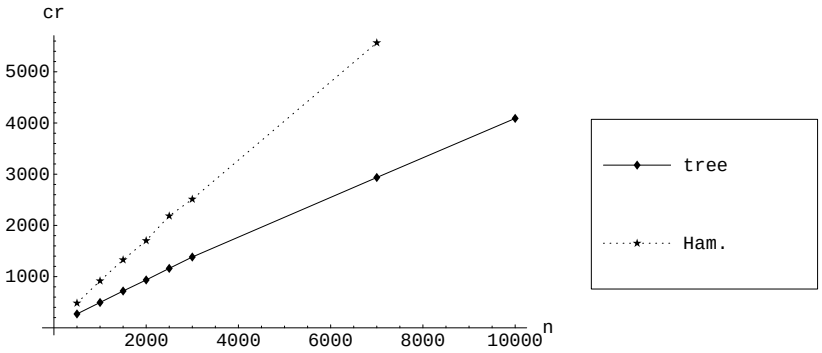

FIGURE 9. The crossing number of the largest diagrammatically prime one component maps generated using 10000 runs. Method two produces maps which are almost twice as large as the knots in method one.

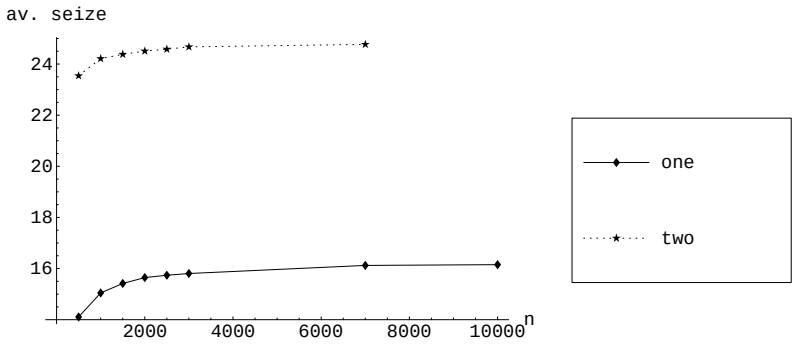

FIGURE 10. The average crossing number of the diagrammatically prime one component maps generated using 10000 runs. Method two produces maps which are on average about 50 percent larger than the maps generated with method one.

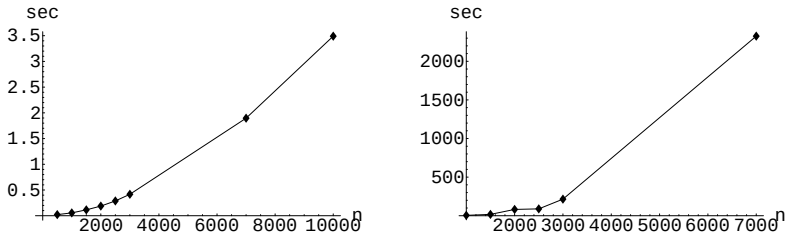

FIGURE 11. The average run time of the two methods in seconds. Method one on the left, method two on the right.

6. Conclusion

One observation is that it is indeed difficult to generate large random diagrammatically prime diagrams effectively. To obtain actual knots from these diagrams will also be a challenging problem. One important issue is: once an assignment of overpass/underpass is made at each crossing of the diagram, what is the relation between the crossing number of the resulting knot and the actual crossings in the diagram? Another algorithm will be needed to carry out such a task. There are many other unresolved issues here. For example, what is the distribution of the diagrams so generated? Is it uniform? Or is it more likely to have certain knot diagrams than others? To summarize: the methods we proposed here can guarantee the following at this point:

1. Any minimum projection diagram of a prime knot can be generated using method one and

2. Hamiltonian projection diagrams of a prime knot can be generated by method two and such diagrams have at most four times as many crossings as the crossing number of the knot.

The data shows that it is possible for one to obtain fair size samples of large random knot diagrams within a reasonable time frame using these methods.

References

[BS] G. Buck and J. Simon, *Thickness and Crossing Number of Knots*, Topology Appl. **91**(3) (1999), pp. 245–257.

[BZ] G. Burde and H. Zieschang, *Knots*, De Gruyter (1985).

[CKS] J. Cantarella, R. B. Kusner and J. M. Sullivan, *Tight Knot Values Deviate from Linear Relations*, Nature, **392** (1998), pp. 237-238.

[D] Y. Diao, *The Lower Bounds of the Lengths of Thick Knots*, J. Knot Theory Ramifications **12** (2003), no. 1, 1–16.

[DE] Y. Diao and C. Ernst, *The Complexity of Lattice Knots*, Topology Appl. **90**, no. 1-3 (1998), pp. 1-9. **12** (2003), no. 1, pp. 1–16.

[DET] Y. Diao, C. Ernst and M. Thistlethwaite, *The Linear Growth in Rope Length of a Family of Knots*, J. Knot Theory Ramifications **12** (2003), no. 5, 709–715.

[DEY] Y. Diao, C. Ernst and X. Yu, *Hamiltonian Knot Projections and Lengths of Thick Knots*, to appear in Topology Appl.

[DT] C. H. Dowker and M. Thistlethwaite, *Classification of knot projections*, Topol. Appl. **16** (1983), pp. 19–31.

[K] R. M. Karp, *Reducibility among combinatorial problems*, In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, Plenum Press, 1972. pp. 85–103.

[RS] E. Rawdon and J. Simon, *The Moebius energy of thick knots*, Topology Appl. **125** (2002), no. 1, pp. 97–109.

[S] P. Schaeffer, *Bojective census and random generation of Eulerian planar maps with prescribed vertex degrees*, preprint- check.

[ST] C. Sundberg, and M. B. Thistlethwaite, *The rate of growth of the number of prime alternating links and tangles*, Pacific J. Math., Vol.182, No.2 (1998), 329–358.

[SZ] P. Schaeffer, and P. Zinn-Justin, *On the asymptotic number of plane curves and alternating knots*, preprint- check.

[T] M. B. Thistlethwaite, *On the structure and scarcity of alternating links and tangles*, J. Knot Theory Ramifications, Vol.7, No.7 (1998), 981–1004.

Department of Mathematics, University of North Carolina at Charlotte, Charlotte, NC 28223
E-mail address: ydiao@uncc.edu

Department of Mathematics, Western Kentucky University, Bowling Green, KY 42101
E-mail address: claus.ernst@wku.edu

Department of Computer Science, Western Kentucky University, Bowling Green, KY 42101
E-mail address: uta.ziegler@wku.edu