Bounds for minimal step number of knots in the simple cubic lattice

R. Scharein[†], K. Ishihara[‡], J. Arsuaga[†], Y. Diao[*], K. Shimokawa[‡] and M. Vazquez[†][§]

[†]Department of Mathematics
San Francisco State University
1600 Holloway Ave
San Francisco, CA 94132, USA.
[‡]Department of Mathematics
Saitama University
Saitama, 338-8570, Japan.
[*]Department of Mathematics and Statistics
University of North Carolina Charlotte
Charlotte, NC 28223, USA.

Abstract. Knots are found in DNA as well as in proteins, and they have been shown to be good tools for structural analysis of these molecules. An important parameter to consider in the artificial construction of these molecules is the minimal number of monomers needed to make a knot. Here we address this problem by characterizing, both analytically and numerically, the minimum length (also called minimum step number) needed to form a particular knot in the simple cubic lattice. Our analytical work is based on an improvement of a method introduced by Diao to enumerate conformations of a given knot type for a fixed length. This method allows to extend the previously known result on the minimum step number of the trefoil knot 3_1 (which is 24) to the knots 4_1 and 5_1 and show that the minimum step numbers for the 4_1 and 5_1 knots are 30 and 34 respectively. We report on numerical results resulting from a computer implementation of this method. We provide a complete list of estimates of the minimum step numbers for prime knots up to 10 crossings, which are improvements over current published numerical results. We enumerate all minimal lattice knots of a given type and partition them into classes defined by BFACF type 0 moves.

AMS classification scheme numbers: Primary 57M25, secondary 92B99.

Submitted to: *J. Phys. A: Math. Gen.*

§ To whom correspondence should be addressed (mariel@math.sfsu.edu)

1. Introduction

Knots are commonly found in long biopolymers such as DNA and proteins. In the laboratory DNA knots are used as probes for characterizing biophysical properties of DNA in solution [28, 30] and in confinement [3]; analyzing enzymatic actions, such as site-specific recombination (e.g. [4, 7, 33]) and topoisomerase action [5, 8, 13, 27, 31]. DNA knots are also engineered as nanotechnological devices [25, 29].

The backbone of some polypeptide chains (i.e. monomeric proteins) from different organisms such as viruses [35], bacteria [22] or humans [34] follow trajectories that are knotted upon joining of the C-terminus with the N-terminus of the protein. Examples of families that contain knotted proteins include RNA methyltransferases [26], kinases [35] and transmembrane proteins [22]. The existence of such protein knots challenges some of the current hypotheses in protein folding and evolution [24, 32, 34]. Understanding the protein knots will give insight into these important processes.

Most theoretical studies of knotted biopolymers deal with properties that are observed when the molecules are long (e.g. [3, 11, 12, 21]). However the formation of knots in short biopolymers is also important in DNA and protein nanotechnology. A question that naturally arises from these studies is the following: What is the minimum number of nucleotides (for DNA) or residues (for proteins) that one needs to form a knot of given knot type?

This question has been addressed by a number of theoretical studies in the continuum ([6]) and in lattices [9]. Here we aim to determine the minimum length needed to construct a knot in the simple cubic lattice ($\mathbb{Z}^3$). This number is called the *minimum step number* of the knot. In [9] it was rigorously shown that the minimum step number for any non-trivial knot in $\mathbb{Z}^3$ is 24 and only a trefoil can be realized with 24 steps. Numerical constructions of various minimal step knots with up to length 60 were studied by Janse van Rensburg [18] and were later extended in [15].

In this paper, we first introduce needed definitions and terminology (section 2). In section 3 we present an improved version of the method developed by Diao in [9]. This refined method proves to be very useful in the quest to find the minimum step number for a given knot in $\mathbb{Z}^3$. We outline how the method can be used to prove that the minimum step number for the 4_1 knot is 30 and that the minimum step number for the 5_1 knot is 34. The complete proofs are quite lengthy and will be reported elsewhere [16, 17]. In section 4 we explain the numerical methods, based on the BFACF algorithm, used to numerically search for the minimum step numbers for prime knots up to 10 crossings. In section 5 we report on the numerical results. Strictly speaking, the numbers obtained this way are only upper bounds of the corresponding minimum step numbers since they have not been analytically proved, except for the knots 3_1, 4_1 and 5_1. However, we expect most of these numbers to be very close to the corresponding minimum step numbers (if not already the minima) due to the power and depth of the search method. In particular, we are able to improve a few previously reported minimum step number bounds for some knot types. Finally, for each prime knot $\mathcal{K}$ up to 10 crossings, we enumerate all numerically found minimal step lattice polygons of knot type $\mathcal{K}$ up to rigid motions. We construct equivalence classes of minimal polygons, where two minimal polygons of type $\mathcal{K}$ are equivalent if they are related by rigid motion. Each class is represented by a suitably chosen *canonical polygon*. Then we enumerate all *canonical polygons* for prime knots up to

10 crossings. We find exactly 75 canonical polygons for the 3_1, some of which were missing in the earlier enumeration [9]. The exact numerical results are given in a series of tables in the Appendix (Tables 4-7). Table 6 and its accompanying figures illustrate the complete list of canonical lattice polygons with 24 steps for the trefoil.

2. Definitions

Consider the simple cubic lattice $\mathbb{Z}^3$. A *step* is a line segment of unit length joining two lattice points in $\mathbb{Z}^3$ (namely points with integer coordinates). A *lattice polygon* K of length n is a closed self-avoiding walk in $\mathbb{Z}^3$ consisting of n steps. The *minimum step number* of a knot type $\mathcal{K}$ is the minimum number of steps needed to construct a lattice polygon K of knot type $\mathcal{K}$. A step parallel to the x-axis is called an x-step; y and z-steps are defined similarly.

Definition 1 *Two lattice polygons are said to be equivalent if they can be superimposed by a rigid motion, i.e., by any combination of translations, rotations and reflections. We call such equivalence class a canonical lattice polygon.*

In section 4 we describe a *standardization algorithm* which allows us to chose a unique polygon to represent each class. By a slight abuse of notation, in this work we refer to the class, and to the standard polygon that represents it, as a *canonical polygon*.

Let K be a lattice polygon of n steps. Let a (resp. b, c) be the number of x-steps (resp. y, z-steps) in K. In the enumeration section we only enumerate canonical lattice polygons of minimum step number. Without loss of generality we assume that $c \geq \max\{a, b\}$. Let $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ be the orthogonal projection in the z-direction, and let G be the planar graph $\pi(K)$ (where the lattice points are the vertices and the steps joining the vertices are the edges). The *multiplicity* $m(e)$ of an edge e in G is the number of steps of K projected onto e by π. The *multiplicity* $m(v)$ of a vertex v in G is the sum of multiplicities of edges incident to v. In the context of this paper G is a weighted graph with the vertex and edge weights defined as their corresponding multiplicities. The sum of all edge multiplicities is called the *total multiplicity* of G and is denoted by m. Notice that $m = a + b \leq \frac{2n}{3}$ when $c \geq \max\{a, b\}$.

Figure 1 below shows a weighted planar graph on the square lattice Z^2 and several possible lattice knots that project onto this graph.

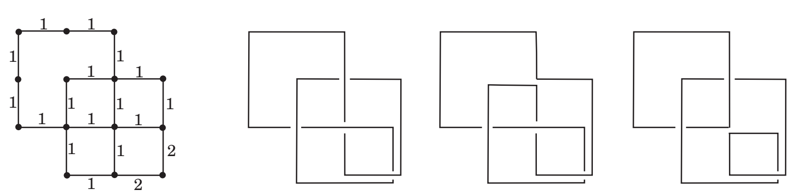


Figure 1. This figure illustrates three possible knot diagrams obtained from the weighted graph G shown on the left. Many knot and link diagrams can be constructed from a single graph G by varying its weight assignments.

The spatial behavior of lattice polygons of fixed knot type and length can be simulated in the computer using the BFACF algorithm ([**?**, 2]). BFACF introduces local moves in a polygon. Figure 2 defines the three basic moves used in the BFACF algorithm, called type 0, 2 and -2 moves, respectively. A type 0 move preserves the step number of the lattice polygon, a type 2 move increases the step number by 2 and a type -2 move decreases the step number by 2. The BFACF algorithm determines a Markov chain which can sample all possible configurations of the polygons of length $L \pm e$ without changing the knot type ([20, 23]). The significance of the BFACF moves is implied in the following theorem.

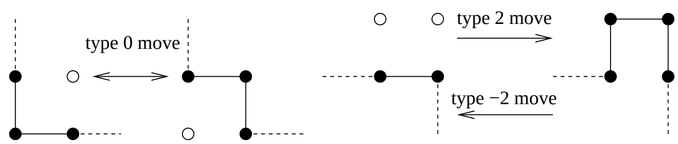


Figure 2. The figures illustrate Type 0, 2 and -2 moves from the BFACF algorithm. A vertex in the figure marked by an open circle indicates a lattice point not occupied by the lattice polygon.

Theorem 1 *[20] Let* K_1 *and* K_2 *be any two lattice polygons of the same knot type. Then* K_2 *can be obtained from* K_1 *by a finite sequence of BFACF moves (i.e., a finite sequence of type* 0, 2 *and* -2 *moves).*

Definition 2 *A lattice polygon* K *is called reducible if its step number can be reduced by applying only type* 0 *and* -2 *moves. Otherwise* K *is called irreducible.*

Note that a lattice polygon with minimum step number is irreducible, but not all irreducible lattice polygons have minimum step number.

3. Theoretical results

In this section we outline the approach used to obtain the theoretical minimum step numbers for knots 4_1 and 5_1.

Theorem 2 *The minimum step number of* 4_1 *is 30 and the minimum step number of* 5_1 *is 34.*

Our approach is a modified and improved version of the original method used in [9] to prove that 24 is the minimum step number of 3_1. The method used in [9] can be summarized as the following algorithm.

Minimal Step Algorithm [9]

Step 1. Enumerate all weighted planar graphs in the square lattice $\mathbb{Z}^2$ with total multiplicities at most $\frac{2n}{3}$. Note that each edge has weight (multiplicity) at least 1 in the weight assignment.

Step 2. For each weighted graph G obtained from Step 1, generate all possible knot diagrams K_G with G as its projection.

Step 3. For each knot diagram K_G enumerate its possible lattice polygon realizations with at most n steps and determine its knot type.

When $n \leq 24$, the enumeration in the above algorithm is still manageable. However, as n increases, the cases that need to be examined grow exponentially. In order to obtain the results in Theorem 2, significant improvements to the above algorithm are needed. Here we will present several partial results to show how the number of calculations in the above algorithm can be reduced. Due to the length and complexity of these proofs only an outline is provided. The full proofs can be found in [17, 16]. First we have the following proposition.

Proposition 1 *Let K be an irreducible lattice polygon with n steps of a nontrivial prime knot type. Then $G = \pi(K)$ is contained in rectangles of size $(\frac{k}{2}+1)\times(\frac{n-2k}{8}+1)$ for an integer k with $2 \leq k \leq \frac{n}{6}$.*

Proposition 1 follows from Proposition 2 below directly.

Proposition 2 *Let K be an irreducible lattice polygon with n steps of nontrivial prime knot type. Then K is contained in boxes of size $(\frac{k}{2}+1)\times(\frac{l}{2}+1)\times(\frac{n-2(k+l)}{4}+1)$ for integers k and l where $2 \leq k \leq \frac{n}{6}$ and $k \leq l \leq \frac{n-2k}{4}$.*

Proof. Let i be an element of $\{x, y, z\}$ and s_i the number of i-steps of K. Note that $s_x + s_y + s_z = n$. Let $D_i(K)$ be the diameter of a lattice polygon K for i-direction, which is the length of the image of projection of K into i-axis.

We claim that $D_i(K) \leq \frac{s_i}{4} + 1$. To see this, let a be a number of the form $k + \frac{1}{2}$ for some $k \in \mathbb{Z}$. Let $P_i(a)$ be the intersection of K and the plane $z = a$ in $\mathbb{R}^3$. If $|P_i(a)| = 2$, then $P_i(a-1) = \emptyset$ or $P_i(a+1) = \emptyset$. Otherwise K is a composite knot or reducible. Hence if $P_i(a-1) \neq \emptyset$ and $P_i(a+1) \neq \emptyset$, then $|P_i(a)| \geq 4$. It follows that $s_i = \sum_a |P_i(a)| \geq 4(D_i(K) - 2) + 4 = 4D_i(K) - 4$, hence $D_i(K) \leq \frac{s_i}{4} + 1$.

Now let $s_x = 2k$ and $s_y = 2l$. Since s_x and s_y are even, k and l are integers. Without loss of generality, let us assume $s_x \leq s_y \leq s_z$. Then $2k = s_x \leq \frac{s_x+s_y+s_z}{3} = \frac{n}{3}$ and $2l = s_y \leq \frac{s_y+s_z}{2} = \frac{(s_x+s_y+s_z)-s_x}{2} = \frac{n-2k}{2}$. Hence we have $k \leq \frac{n}{6}$ and $k \leq l \leq \frac{n-2k}{4}$. If $k = 1$, then K is 1-bridge knot. Hence it is trivial. Therefore $2 \leq k \leq \frac{n}{6}$. By Claim, $D_x(K) \leq \frac{s_x}{4}+1 = \frac{k}{2}+1$, $D_y(K) \leq \frac{s_y}{4}+1 = \frac{l}{2}+1$, and $D_z(K) \leq \frac{s_z}{4}+1 = \frac{n-2(k+l)}{4}+1$. Hence K is contained in a box in $\mathbb{R}^3$ with size $(\frac{k}{2}+1)\times(\frac{l}{2}+1)\times(\frac{n-2(k+l)}{4}+1)$. □

With Proposition 1, we can add a new step to the **Minimal Step Algorithm** prior to Step 1, and modify Step 1 accordingly as shown below. These two new steps greatly reduce the total number of cases to be examined.

Step 0. Enumerate all rectangles in $\mathbb{Z}^2$ whose dimensions satisfy the conditions given in Proposition 1.

Step 1′. For each rectangle R identified in Step 0, enumerate all weighted planar graphs in R with total multiplicities at most $\frac{2n}{3}$.

The following proposition allows to identify graphs G that produce reducible lattice polygons.

Proposition 3 *[17] Let e be an edge of G whose end points are v_1 and v_2. If $2m(e) \geq m(v_1)$ and $2m(e) > m(v_2)$ then K is reducible.*

Proposition 4 *[17] Let $c(K)$ be the crossing number of K. Then*

$$c(K) \leq \sum_{v \in G} T(\frac{m(v)}{2}) - \sum_{e \in G} T(m(e)) + 3(f-1)$$

where $T(x) = \frac{(x-2)(x-3)}{2}$ and f is the number of faces of G.

Proposition 4 can be used to estimate the crossing number of the lattice polygon K in $\mathbb{Z}^3$ using the multiplicities and the number of faces of G. For example, the graph in Figure 1 can be a projection of the figure eight knot. The right hand side of the inequality in Proposition 4 for that graph is 4 (the estimation in Proposition 4 is sharp in this case), therefore that graph cannot be a projection of 5_1 in $\mathbb{Z}^3$. This fact is used in the proof of Theorem 2. Due to length consideration, the proofs for Propositions 3 and 4 are not presented here. Instead we provide an example in Figure 3 to help with understanding Proposition 3.

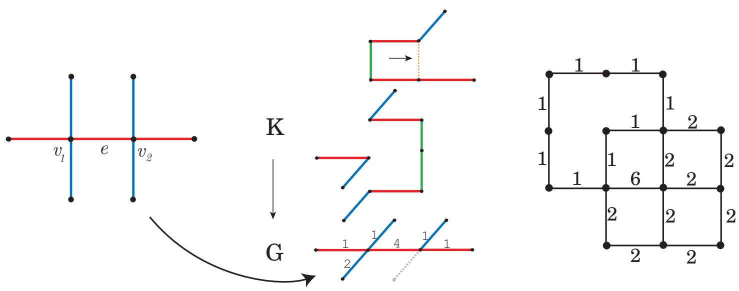


Figure 3. Left: A weighted graph G with an edge e spanning two vertices v_1 and v_2. Center: An example of e with $m(e) = 4$, $m(v_1) = 8$ and $m(v_2) = 6$. In this example, e, v_1 and v_2 satisfy the conditions in Proposition 3. This figure shows a 3-dimensional conformation K projecting onto G, and illustrates the reducibility of K. Right: Another graph G whose central edge e has $m(e) = 6$ and satisfies proposition 3. Graphs with this property do not need be considered in the construction of minimum step lattice knots.

We end this section with an outline of the proof of Theorem 2: For the case of 4_1, one can construct 4_1 in $\mathbb{Z}^3$ with 30 steps as shown in Figure 4. Thus we only need to show that any lattice polygon with at most 28 steps is either the trivial knot or the trefoil knot. By [9, 10], it is known that any lattice polygon with at most 24 steps is trivial or the trefoil knot. Let K be an irreducible non-trivial lattice polygon with $n = 26$ or 28 steps. By taking a suitable projection one obtains a graph G with total multiplicity at most 18. By Proposition 1, G is contained in a rectangle of dimension 2×4 or 3×3. By using the Euler characteristic, one can show that there are at most 6 faces in G. We then apply Propositions 2, 3 and 4 to further simplify the enumeration of these graphs. Finally, we follow the steps proposed in [9, 10] to list corresponding lattice polygons and determine their knot types and show that each lattice polygon is either trivial or the trefoil.

The case of 5_1 follows the same steps but the calculations are more involved.

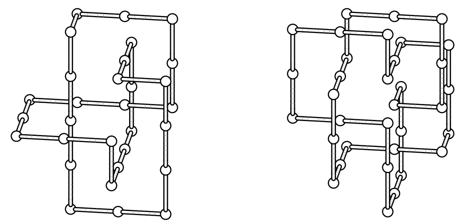


Figure 4. Realizations of 4_1 and 5_1 on the lattice with 30 and 34 steps.

4. Simulation methods

In our simulations we used the BFACF algorithm as described in [23]. The BFACF algorithm has been used to estimate properties of knots in $\mathbb{Z}^3$ such as the writhe, the radius of gyration, and the minimum stick number[18, 19]. This algorithm introduces local moves in a lattice polygon which, when iterated for long periods of time, simulate spatial movement. The BFACF algorithm can sample all possible configurations of the polygons within a given knot type [23]. That is, the BFACF algorithm generates Markov chains whose ergodicity classes are the knot types [20]. The algorithm samples conformations from a Gibbs distribution with one adjustable parameter z, the fugacity per bond, that satisfies that $0 \leq z \leq z_c$ where $z_c = 0.2134$ is the inverse of the connective constant in $\Re^3$ ([23]). The resulting Markov chain is generated using the three elementary moves defined earlier in Figure 2. Type 0, +2 or −2 moves are selected with probabilities $p(0)$, ($p(2)$ and $p(-2)$, respectively. The probabilities are given by $p(0) = \frac{1+z^2}{2(1+3z^2)}$, $p(2) = \frac{z^2}{1+3z^2}$ and $p(-2) = \frac{1}{1+3z^2}$.

To ensure that duplicated lattice polygons are not counted in our enumeration process, we introduce a method to place a lattice polygon (of minimum steps) in "standard form". We represent a lattice polygon by a `NEWSUD` *string*, namely a sequence of letters from the set {N, E, W, S, U, D} (abbreviations for North, East, West, South, Up and Down) with each letter representing the corresponding edge direction.

Standardization algorithm

Start from a given lattice polygon in the `NEWSUD` format, for example:

`UENNWWSWDDEEUUUNWWDDESSE`

(a) List all 24 forms obtained from it by rotations in $\mathbb{Z}^3$,

```
UENNWWSWDDEEUUUNWWDDESSE
SEUUWWDWNNEESSSUWWNNEDDE
DESSWWNWUUEEDDDSWWUUENNE
          ⋮
USEENNWNDDSSUUUENNDDSWWS
UENNWWSWDDEEUUUNWWDDESSE.
```

(b) Because these forms are computer generated, they have a starting edge and an orientation, both of which are not interesting for the purposes of our enumeration. Therefore for each rotated form, we list all possible shifts,

`UENNWWSWDDEEUUUNWWDDESSE`

```
EUENNWWSWDDEEUUUNWWDDESS
SEUENNWWSWDDEEUUUNWWDDES
            ⋮
```

(c) For each rotated or shifted form we also consider the form with the opposite (reverse) orientation. Note that this is not simply the reversal of the string since that would produce the mirror image of the lattice polygon. The lattice polygon `NNDDSEDDWWUUUEENDDWSSUUU` reversed is `DDDNNEUUSWWDDDEEUUWNUUSS` (string reversed and substitutions `N` $\leftrightarrow$ `S`, `E` $\leftrightarrow$ `W` and `U` $\leftrightarrow$ `D` made). Furthermore, in the case of achiral knots, the additional $2 \times n \times 24$ strings resulting from reflections are also considered.

(d) For an n step lattice polygon this algorithm produces up to $4 \times n \times 24$ forms (many of which may be duplicates). We choose the lexicographically least of all these as the *standard form*. For example, the standard form of `UENNWWSWDDEEUUUNWWDDESSE` is `DDDEEUUWSWWNNEDESSEUUWWN`.

Conformations of lattice polygons of the same knot type are different if their standard forms are different. Notice that it is ideal to use this standard form to represent the class of polygons modulo rigid motion introduced in Definition 1. We refer to the polygon in standard form as a *canonical polygon*.

The numerical simulation involves two major phases, as follows:

Phase 1: Identification and standardization of minimum step number lattice polygons. Each knot type is subjected to several iterations of the following numerical search.

(i) Run the BFACF algorithm on a lattice polygon of fixed knot type, periodically switching between high and low z values. The lattice polygon with the fewest edges produced in this process is retained.

(ii) If the number of edges of the lattice polygon found in the step (i) is greater than the current lower bound for the step number of the knot type, discard this lattice polygon and go on to the next knot type.

(iii) If the lattice polygon is not discarded, it is then put through a *standardization procedure*. From this procedure results a canonical polygon. If the canonical polygon has not been found before, it is added to the list as a new canonical lattice polygon.

Notice that as the list of distinct conformations for a given knot type grows during the random search using the identification algorithm described above, only canonical conformations obtained form the standardization algorithm are stored. Incoming potentially new conformations are standardized first and then added to the list if they are not already there.

Phase 2: Expansion.

The second phase expands the data found during phase 1. Each conformation is exhaustively run through a modified BFACF algorithm allowing only type-0 moves. Only those canonical conformations not already found after a standardization step are retained. For most knots up to eight crossings, this phase produces very few or no additional conformations. However the expansion phase is necessary for some nine and ten crossing knots.

In order to better group the canonical lattice polygons we define a new equivalence class induced by BFACF type 0 moves.

Definition 3 *Two canonical lattice polygons are* type 0 equivalent *if one can be taken into the other via a finite sequence of BFACF type 0 moves.*

Definition 4 *A canonical lattice polygon is* taut *if we cannot apply any type* 0 *move to any of its lattice polygons.*

In other words, a canonical lattice polygon is taut if its type 0 equivalence class contains only one element. In the next section we compute the number of type 0 equivalence classes under this relation for each knot type.

5. Numerical results

We ran the numerical simulations following the identification, Standardization and expansion phases as explained in the previous section. To obtain the results reported here, a z value was chosen uniformly at random in the interval $(0, 0.2)$ ("high" z). The simulation was run for $100,000$ iterations, followed by a run of 60,000 iterations at $z = 0.02$ ("low" z). This cycle was repeated 100 times. Note that the actual values and length of the run between z-changes was quite arbitrary.

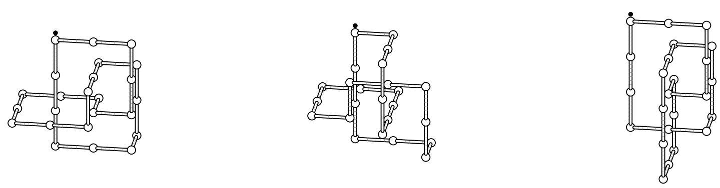
Due to their length, tables summarizing the numerical results are placed in the appendix. Table 4 outlines our main numerical results for all prime knots up to 10 crossings. The knot types are given in Column **K**, the minimum step number bounds are given in Column **S**, the numbers of canonical conformations found in Phase 1 are given in Column **R**, the numbers of conformations found in Phases 1 and 2 combined are given in Column **E** and the numbers of type 0-equivalent classes are given in Column **C**. In the following subsections we will discuss our findings. Numerical estimates of minimal step number, as well as examples in the `NEWSUD` form are provided in the appendix (after Table 7) for each prime knot with 10 crossings or less.

5.1. The case of the trefoil knot revisited

We found 75 canonical lattice polygons for the trefoil knot, partitioned into three type 0 equivalence classes. Table 6 in the appendix lists all canonical lattice trefoils: canonical lattice polygons $1-1$ to $1-18$ are in the first equivalence class; polygons$2-1$ to $2-56$ are in the second equivalence class; $3-1$ is in the third. The canonical lattice polygon $3-1$ is taut. Using the algorithm proposed in [9], Diao enumerated all lattice polygons with 24 steps which are trefoil knots (in [10]: see p.1253, Figure 8). It is easily verified that each case in Diaos enumeration is contained in our 75 canonical lattice polygons. See the data in the appendix for details (table 6 and its accompanying figures). However, we found that some lattice polygons in [10] were counted twice (for instance one lattice polygon in Figure 8(c) can be obtained from one in Figure 8(f) after a suitable rotation is applied) and some were completely missed. More precisely, there are 27 canonical lattice polygons that were not counted in [10]. These are listed in the following table and three of them are illustrated in Figure 5. A complete list of these 75 canonical lattice polygons and their geometric representations is given in Table 6, and on the figures below the table, in the appendix.

1-1	DDDEENUUWSSDWWNNEESEUUWW	1-2	DDDEENUUWSSWDWNNEESEUUWW
1-3	DDDEENUUWSSWWDNNEESEUUWW	1-4	DDDEENUUWSSWWNDNEESEUUWW
1-5	DDDEENUUWSSWWNNDEESEUUWW	1-6	DDDEENUUWSSWWNNEDESEUUWW
2-1	DDDEESUUWWDWNNEESSSUUNNW	2-6	DDDEESUUWWWDNNEESSSUUNNW
2-7	DDDEESUUWWWDNNEESSSUUNWN	2-11	DDDEESUUWWWDNNEESSWSUUNN
2-12	DDDEESUUWWWNDNEESSSUUNWN	2-16	DDDEESUUWWWNDNEESSWSUUNN
2-17	DDDEESUUWWWNNDEESSSUUWNN	2-21	DDDEESUUWWWNNEEDSSSWUUNN
2-22	DDDEESUUWWWNNEEDSSWSUUNN	2-23	DDDEESUUWWWNNEESDSWSUUNN
2-24	DDDEEUSUWWDWNNEESSSUUNNW	2-28	DDDEEUSUWWWDNNEESSSUUNNW
2-29	DDDEEUSUWWWDNNEESSSUUNWN	2-32	DDDEEUSUWWWDNNEESSWSUUNN
2-33	DDDEEUSUWWWNDNEESSSUUNNW	2-35	DDDEEUSUWWWNDNEESSWSUUNN
2-48	DDDEEUUSWWWDNNEESSWSUUNN	2-49	DDDEEUUSWWWNDNEESSWSUUNN
2-50	DDDEEUUSWWWNNEEDSSSUUNNW	2-54	DDDEEUUWSWWDNNEESSWSUUNN
3-1	DDDEENUUWSSDDDNNUUSEUUWW		

Table 1. The 27 canonical lattice polygons missing from [10].



(1-1) (2-1) (3-1)

Figure 5. Standard lattice polygons of 1-1, 2-1 and 3-1 as listed in Table 1. Their projection graphs (as weighted plane graphs) were not recorded in [10].

Since each of the 75 canonical lattice polygons has 24 rotations, one may expect $75 \times 24 = 1800$ different lattice polygons. However, some of the canonical lattice polygons exhibit geometric symmetries. For example some rotated versions overlap (i.e. are not distinct lattice polygons). Eliminating such duplicates results in 1664 distinct lattice polygons. Taking reflections into account, we obtain $2 \times 1664 = 3328$ lattice polygons. We thus report that the total number of minimum step number lattice polygons should be 3328, not 3496 as reported in [10].

5.2. The cases of 4_1 *and* 5_1 *knots*

Our numerical results confirm the analytical results of [17, 16]. That is, the 4_1 and 5_1 knots have minimum step numbers of 30 and 34 respectively. It is interesting to note that 4_1 has only 76 minimum step number canonical lattice polygons while 5_1 has twice as many canonical lattice polygons (142) as 3_1 does.

5.3. Improvement over published data

Our calculations offer improvements over some of the minimum step bounds proposed in [18] and in [15]. The following knots were found to have lattice polygon representations of 50 steps, improving the 52 reported in [18]: 8_8, 8_{10}, 8_{11} and 8_{14}. A minimum step number bound of 46 for 8_{20} was reported in [15], we improved that to 44.

A list of estimations on the total number of knot types that a lattice polygons with a given step number can realize was given in [18]. The following table shows an updated list based on our data.

s	4	24	30	34	36	40	42	44	46
n	1	2	3	4	5	8	9	14	18
s	48	50	52	54	56	58	60	62	64
n	22	40	49	74	119	130	175	244	250

Table 2. Numerical estimates on n, the number of prime knot types realizable by a lattice polygon with step number up to s.

5.4. Further observations

It is interesting to see the large range for the number of canonical lattice polygons within knots with the same crossing numbers. For instance, in the eight crossing family of knots, our search found only one realization for 8_3 and 8_7, while for other knots like 8_2, we found almost 2000. This range seems to grow rapidly. For example, we only found one canonical lattice polygon for 9_{31}, 10_{72} and 10_{123}. On the other hand, we found 152374 canonical lattice polygons for 10_{140}!

Table 3 below summarizes our numerical finding on the relationship between the class of knots with a given crossing number and the range of step numbers needed for the lattice polygons to realize these knots. This limited study seems to indicate that the non-alternating knots generally require less steps in their lattice polygon realizations.

Crossing number	Alternating	Non-alternating
3	24	-
4	30	-
5	34-36	-
6	40	-
7	44-46	-
8	48-52	42-46
9	54-58	48-52
10	58-64	48-58

Table 3. Crossing numbers and minimum step numbers of prime knots

5.5. Minimum stick numbers via the canonical lattice polygons

We also studied the minimum stick number of a knot using its corresponding canonical lattice polygons. The (lattice) *stick number* of a knot type $\mathcal{K}$ is the number of (straight) edges, not necessarily of unit length, required to construct a representative of $\mathcal{K}$ in $\mathbb{Z}^3$. An estimate of the stick number of knot type $\mathcal{K}$ can be obtained using the canonical lattice polygons of $\mathcal{K}$ described above, by simply counting the number of times the lattice polygon of the knot changes direction in all of the NEWSUD strings for a given knot. There exist minimum stick representatives of a knot that do not correspond to canonical polygons (of minimum step) when their edges are subdivided into edges of unit length. For knots up to eight crossings (plus a few others), we ran a separate experiment. We set the BFACF z parameter randomly within the range $(.13, .19)$, performed 100000 iterations of the BFACF algorithm, then performed all possible type (-2) moves and finally counted the number of direction changes. These steps were repeated until a candidate was found or a time limit (several hours) was reached. This method produced new minimum stick candidates for the knot types 5_1, 5_2, 6_1, 6_3, 7_2, 7_3, 7_4, 7_5, 7_6, 7_7, 8_2, 8_3, 8_4, 8_5, 8_7, 8_{19} and 10_{27}. Our numerical results also confirm the known theoretical results for 3_1 and 4_1 (12 and 14 respectively) shown in [14]. A complete list of our findings on the stick numbers is given in Table 5 in the appendix.

6. Conclusion

Knots in biopolymers, such as closed DNA molecules and open polypeptide chains (proteins) can be engineered in the laboratory. In this context it is natural to ask what is the minimum number of monomers (nucleotides for DNA and residues for proteins) needed to form a given knot. We have addressed this question using the simple cubic lattice as a model. We propose new analytical tools that improve current published results in the sense that the process of finding the minimum step number is made more efficient. Our data includes new minimal step number and minimal stick number bounds. These improvements allowed us to determine the minimum step number for 4_1 and 5_1. We have improved some current analytical and numerical results in the estimations on the number of canonical lattice polygons of a given knot type. For instance we have numerically found that there are a total of 75 different canonical lattice polygons that are trefoils, some of which were not counted before. Our numerical study was broad enough to cover all prime knots up to 10 crossings. However, we need to point out that our results (analytical and numerical) are obtained by enumerating all possible lattice polygons (of the given knot type), it will be difficult to use the same method to generalize these results to knots that exhibit more realizations (as shown by our simulations), such as the 5_2 knot, which has 2362 canonical lattice polygons. More powerful analytical results will be needed in order to reduce the possibilities in the enumeration process.

Acknowledgments

Support for this work came from NSF grant DMS-0712958 (Y.D.), NIH grants 2S06GM052588 (M.V and R.S.), NIH 2S06GM52588-12 (J.A. and R.S.). Additional support came from the Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Scientific Research (C) 18540069 (K.S.). K. Ishihara is partially supported by a Fellowship from the Japanese Society for the Promotion of Science for Japanese Junior Scientists.

References

[1] C. Arãgao de Carvalho and S. Caracciolo 1983, *J Phys (Paris)* **44** pp. 323-331.
[2] C. Arãgao de Carvalho, S. Caracciolo and J. Frölich 1983, *Nucl Phys B* **215** pp. 209-248.
[3] Arsuaga J et al 2002, *Proc Natl Acad Sci USA* **99** 5373–7.
[4] Buck D and Flapan E 2007, *J Mol Biol* **374**(5) 1186- -99.
[5] Buck G R and Zechiedrich E L 2004, *J Mol Biol* **340**(5) 933–9.
[6] Calvo J A and Millett K C 1998, *Ideal Knots: Series of knots and everything* (Ed. Stasiak A, Katrich V and Kauffman L H) **19** 107-28.
[7] Darcy I K et al 2006, *BMC Bioinformatics* **7** 435.
[8] Dean F B et al 1985, *J Biol Chem* **260** 4975–83.
[9] Diao Y 1993, *J Knot Theory Ramifications* **2**(4) 413–25.
[10] ——1994, *J Statist Phys* **74** 1247–54.
[11] ——1995, *J Knot Theory Ramifications* **4**(2) 189–96.
[12] ——, Nardo J and Sun Y 2001, *J Knot Theory Ramifications* **10**(4) 597–607.
[13] Hsieh T 1983, *J Biol Chem* **258** 8413–20.
[14] Huh Y and Oh S 2005, *J Knot Theory Ramifications* **14**(7) 859–67.
[15] Hua X et al 2007, *Topology Appl* **154** 1381–97.
[16] Ishihara K 2009, preprint.
[17] Ishihara K, Shimokawa K and Yamaguchi Y 2009, preprint.
[18] Janse van Rensburg E J 1998, *Ideal Knots: Series of knots and everything* (Ed. Stasiak A, Katrich V and Kauffman L H) **19** 88–106.
[19] E.J. van Rensburg, D.W. Sumners and S.G. Whittington 1998, *Ideal Knots: Series of knots and everything* (Ed. Stasiak A, Katrich V and Kauffman L H) **19** 70–87.
[20] E. J. van Rensburg and S. G. Whittington 1991, *The BFACF algorithm and knotted polygons. J Phys A Math Gen* **24** 5553–67.
[21] Klenin K V et al 1988, *J Biomol Struct Dyn* **5** 1173–85.
[22] King N P, Yeates E O and Yeates T O 2007, *J Mol Biol* **373**(1) 153–66.
[23] N. Madras and G. Slade,*The Self-Avoiding Walk* Probability and Its Applications, Birkhauser, Boston 1996.
[24] Mallam A L et al 2008, *Mol Cell* **30** 642–8.
[25] Mueller J E, Du S M and Seeman N C 1991, *J Am Chem Soc* **113** 6306–8.
[26] Nureki O et al 2002, *Acta Crystallogr D Biol Crystallogr* **58**(7) 1129–37.
[27] Petrushenko Z M et al 2006, *J Biol Chem* **281**(8) 4606–15.
[28] Rybenkov V V, Cozzarelli N R and Vologodskii A V 1993, *Proc Natl Acad Sci USA* **90**(11) 5307–11.
[29] Seeman N C 2003, *Biochemistry* **42** 7259–69.
[30] Shaw S Y and Wang J C 1993, *Science* **260** 533- -6.
[31] Stray J E et al 2005, *J Biol Chem* **280**(41) 34723–34.
[32] Taylor W R 2000, *Nature* **406** 916–9.
[33] Vazquez M, Colloms S and Sumners D W, *J Mol Biol* **346** 493–504.
[34] Virnau P, Mirny L A and Kardar M 2006, *PLoS Comput Biol* **2**(9) e122.
[35] Yeates T O, Norcross T S and King N P 2007, *Curr Opin Chem Biol* **11**(6) 595–603.

Appendix

Table 4: The knot types are given in Column **K**, the minimum step number bounds are given in Column **S**, the numbers of conformations found in Phase 1 are given in Column **R**, the total numbers of conformations found in Phase 1 and Phase 2 are given in Column **E** and the numbers of type 0-equivalent classes are given in Column **C**.

K	S	R	E	C	K	S	R	E	C	K	S	R	E	C
0_1	4	1	1	1	9_{49}	52	751	751	21	10_{84}	62	549	549	35
3_1	24	75	75	3	10_1	60	7512	9649	76	10_{85}	60	25	25	4
4_1	30	76	76	2	10_2	60	9549	18152	266	10_{86}	60	3	3	1
5_1	34	142	142	7	10_3	58	5	5	1	10_{87}	62	1767	1885	56

K	**S**	**R**	**E**	**C**	**K**	**S**	**R**	**E**	**C**	**K**	**S**	**R**	**E**	**C**
5_2	36	2396	2396	28	10_4	60	2550	2578	86	10_{88}	62	1833	1858	19
6_1	40	129	129	5	10_5	58	10	10	2	10_{89}	62	1255	1255	26
6_2	40	684	684	18	10_6	60	45	45	1	10_{90}	64	15416	61324	347
6_3	40	74	74	6	10_7	62	13560	44197	588	10_{91}	62	4917	5435	53
7_1	44	712	712	31	10_8	60	3096	3213	61	10_{92}	62	183	183	8
7_2	46	6941	7011	111	10_9	60	3429	3644	72	10_{93}	62	1528	1542	37
7_3	44	10	10	1	10_{10}	60	1744	1767	85	10_{94}	62	1797	1797	41
7_4	44	4	4	2	10_{11}	60	101	101	3	10_{95}	62	76	76	8
7_5	46	197	197	8	10_{12}	60	277	277	10	10_{96}	64	3923	4202	67
7_6	46	709	709	19	10_{13}	62	172	172	5	10_{97}	62	6	6	1
7_7	44	11	11	3	10_{14}	60	12	12	4	10_{98}	62	26	26	2
8_1	50	495	495	13	10_{15}	60	235	235	7	10_{99}	62	54	54	5
8_2	50	1910	1910	58	10_{16}	60	40	40	4	10_{100}	62	652	652	12
8_3	48	1	1	1	10_{17}	58	5	5	1	10_{101}	62	42	42	5
8_4	50	997	997	32	10_{18}	60	21	21	3	10_{102}	62	1146	1148	16
8_5	50	24	24	4	10_{19}	60	1299	1306	42	10_{103}	62	4603	4918	23
8_6	50	230	230	2	10_{20}	60	82	82	4	10_{104}	62	721	721	12
8_7	48	1	1	1	10_{21}	60	291	291	15	10_{105}	62	67	67	4
8_8	50	65	65	3	10_{22}	60	715	715	7	10_{106}	62	1055	1056	25
8_9	50	735	735	8	10_{23}	60	787	787	11	10_{107}	62	37	37	11
8_{10}	50	35	35	1	10_{24}	62	2770	2917	35	10_{108}	60	161	161	14
8_{11}	50	4	4	2	10_{25}	62	1351	1422	29	10_{109}	64	3829	4062	108
8_{12}	52	54	54	2	10_{26}	62	10374	22450	136	10_{110}	62	4	4	2
8_{13}	50	544	544	22	10_{27}	60	35	35	4	10_{111}	62	221	221	24
8_{14}	50	15	15	2	10_{28}	60	155	155	10	10_{112}	60	20	20	2
8_{15}	52	1674	1676	37	10_{29}	62	926	926	50	10_{113}	60	2	2	2
8_{16}	50	2	2	1	10_{30}	60	35	35	2	10_{114}	62	896	900	26
8_{17}	52	1108	1108	37	10_{31}	60	5	5	1	10_{115}	62	10	10	3
8_{18}	52	74	74	2	10_{32}	62	6631	10037	184	10_{116}	62	641	641	10
8_{19}	42	304	304	14	10_{33}	60	1184	1190	12	10_{117}	62	93	93	9
8_{20}	44	5	5	1	10_{34}	60	115	115	5	10_{118}	64	2296	2330	67
8_{21}	46	1168	1168	10	10_{35}	62	271	271	16	10_{119}	62	28	28	2
9_1	54	6899	7211	149	10_{36}	60	320	320	18	10_{120}	64	358	358	18
9_2	56	33673	68360	564	10_{37}	62	4844	6106	30	10_{121}	60	40	40	3
9_3	54	105	105	3	10_{38}	62	3349	3484	43	10_{122}	62	1197	1197	12
9_4	54	26	26	6	10_{39}	62	3408	3549	73	10_{123}	60	1	1	1
9_5	54	35	35	3	10_{40}	62	1082	1093	34	10_{124}	48	16	16	2
9_6	56	165	165	9	10_{41}	62	7006	10021	77	10_{125}	54	1659	1668	89
9_7	56	3670	3670	33	10_{42}	60	42	42	6	10_{126}	54	9	9	2
9_8	56	2437	2438	66	10_{43}	62	3342	3597	64	10_{127}	56	3569	3905	38
9_9	56	251	251	12	10_{44}	60	266	266	19	10_{128}	54	9578	15296	137
9_{10}	56	2655	2655	16	10_{45}	60	31	31	5	10_{129}	56	13839	90374	156
9_{11}	56	1274	1274	27	10_{46}	60	331	331	16	10_{130}	54	63	63	4
9_{12}	56	826	826	29	10_{47}	60	14	14	2	10_{131}	56	151	151	8
9_{13}	56	697	697	23	10_{48}	60	30	30	4	10_{132}	52	404	404	20
9_{14}	54	71	71	5	10_{49}	62	6744	11000	134	10_{133}	54	1258	1261	15
9_{15}	58	30245	50726	384	10_{50}	62	4666	6141	97	10_{134}	56	9563	14794	142
9_{16}	56	205	205	16	10_{51}	62	592	592	13	10_{135}	58	16627	109823	381
9_{17}	56	2607	2607	68	10_{52}	62	4332	5258	100	10_{136}	54	558	558	25
9_{18}	58	11071	11495	163	10_{53}	62	2273	2500	30	10_{137}	56	5758	8108	62
9_{19}	54	20	20	1	10_{54}	60	50	50	1	10_{138}	56	20	20	1
9_{20}	56	4689	4771	65	10_{55}	62	1226	1226	14	10_{139}	50	335	335	11
9_{21}	56	477	477	9	10_{56}	62	899	902	18	10_{140}	54	17772	152374	247
9_{22}	56	361	361	14	10_{57}	62	268	268	10	10_{141}	54	917	918	11
9_{23}	58	11247	11736	122	10_{58}	62	187	187	14	10_{142}	54	69	69	6
9_{24}	56	422	422	16	10_{59}	62	1163	1164	43	10_{143}	54	415	415	3
9_{25}	58	10689	11245	162	10_{60}	62	1778	1866	36	10_{144}	58	16600	142772	294

K	**S**	**R**	**E**	**C**	**K**	**S**	**R**	**E**	**C**	**K**	**S**	**R**	**E**	**C**
9_{26}	54	69	69	7	10_{61}	60	126	126	5	10_{145}	52	276	276	5
9_{27}	56	1429	1429	68	10_{62}	60	80	80	2	10_{146}	56	4450	5659	47
9_{28}	56	105	105	12	10_{63}	62	2360	2387	47	10_{147}	54	168	168	1
9_{29}	56	8	8	2	10_{64}	62	10607	24697	106	10_{148}	56	5308	8624	49
9_{30}	56	93	93	9	10_{65}	62	5238	6960	88	10_{149}	56	1295	1322	28
9_{31}	54	1	1	1	10_{66}	62	5080	8675	91	10_{150}	56	5981	7969	39
9_{32}	56	228	228	10	10_{67}	62	1050	1050	17	10_{151}	56	251	251	10
9_{33}	56	122	122	8	10_{68}	62	7478	12093	89	10_{152}	54	414	414	15
9_{34}	56	6	6	2	10_{69}	60	2	2	1	10_{153}	54	768	768	3
9_{35}	56	847	847	28	10_{70}	62	429	429	16	10_{154}	56	1502	1503	5
9_{36}	56	310	310	10	10_{71}	62	141	141	14	10_{155}	54	530	530	3
9_{37}	56	97	97	4	10_{72}	60	1	1	1	10_{156}	56	3292	3386	51
9_{38}	58	44	44	6	10_{73}	60	30	30	4	10_{157}	54	2	2	2
9_{39}	58	5990	6066	60	10_{74}	62	4588	5608	126	10_{158}	56	11	11	4
9_{40}	54	7	7	1	10_{75}	62	1216	1216	43	10_{159}	56	4002	4644	15
9_{41}	54	3	3	2	10_{76}	62	4285	4686	30	10_{160}	56	14947	46794	161
9_{42}	48	578	578	16	10_{77}	62	5290	6711	144	10_{161}	50	18	18	1
9_{43}	50	6477	6531	30	10_{78}	62	1612	1621	78	10_{162}	56	32	32	5
9_{44}	50	3653	3653	25	10_{79}	62	721	721	5	10_{163}	56	612	612	12
9_{45}	52	11067	13609	92	10_{80}	62	319	319	4	10_{164}	56	467	467	13
9_{46}	50	42892	124406	136	10_{81}	64	8537	16708	150	10_{165}	56	44	44	2
9_{47}	50	285	285	2	10_{82}	60	7	7	1					
9_{48}	52	1903	1903	14	10_{83}	62	3108	3413	125					

Table 5: The knot types are given in Column **K**, the minimum stick number bounds are given in Column **S**, the numbers of type 0-equivalent classes are given in Column **C**. A superscript ^{k} on a number in Column **S** is the difference between the minimum stick number found using only the minimum step number candidates and the minimum stick number produced by the separate search outlined in Subsection 5.5.

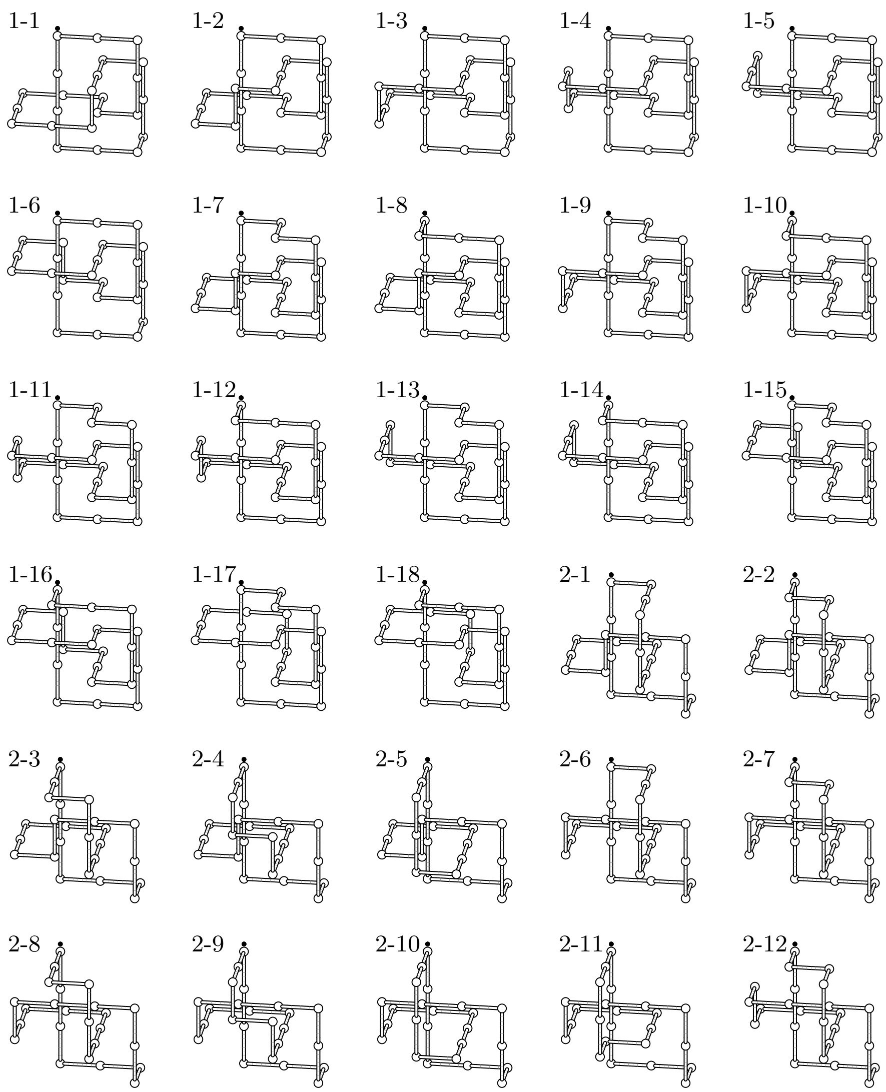
K	**S**	**C**	**K**	**S**	**C**	**K**	**S**	**C**	**K**	**S**	**C**	**K**	**S**	**C**
0_1	4	1	9_{15}	24	15	10_{16}	29	8	10_{66}	24	2	10_{116}	25	24
3_1	12	7	9_{16}	23	1	10_{17}	27	2	10_{67}	26	4	10_{117}	26	8
4_1	14	6	9_{17}	21	21	10_{18}	27	2	10_{68}	24	11	10_{118}	24	10
5_1	17^{1}	2	9_{18}	24	6	10_{19}	25	17	10_{69}	27	1	10_{119}	25	2
5_2	17^{1}	50	9_{19}	25	4	10_{20}	29	8	10_{70}	26	6	10_{120}	25	1
6_1	19^{1}	3	9_{20}	22	6	10_{21}	26	12	10_{71}	28	6	10_{121}	24	2
6_2	17	6	9_{21}	25	8	10_{22}	26	4	10_{72}	27	1	10_{122}	21	1
6_3	18^{1}	9	9_{22}	24	12	10_{23}	25	14	10_{73}	26	3	10_{123}	20	1
7_1	21^{1}	3	9_{23}	24	2	10_{24}	27	8	10_{74}	25	6	10_{124}	22	6
7_2	21^{1}	38	9_{24}	24	6	10_{25}	26	18	10_{75}	24	9	10_{125}	24	18
7_3	22^{1}	2	9_{25}	24	6	10_{26}	25	6	10_{76}	28	16	10_{126}	27	4
7_4	20^{1}	1	9_{26}	23	1	10_{27}	28^{1}	8	10_{77}	27	22	10_{127}	24	1
7_5	22^{1}	4	9_{27}	22	18	10_{28}	24	6	10_{78}	26	8	10_{128}	24	10
7_6	20^{1}	9	9_{28}	25	5	10_{29}	28	12	10_{79}	27	16	10_{129}	23	2
7_7	19^{1}	3	9_{29}	26	2	10_{30}	28	4	10_{80}	28	18	10_{130}	24	3
8_1	23	12	9_{30}	23	4	10_{31}	27	2	10_{81}	26	44	10_{131}	24	2
8_2	21^{1}	12	9_{31}	24	1	10_{32}	24	4	10_{82}	27	2	10_{132}	21	1
8_3	24^{1}	1	9_{32}	23	1	10_{33}	25	6	10_{83}	23	1	10_{133}	24	16
8_4	22^{1}	6	9_{33}	22	2	10_{34}	28	8	10_{84}	24	3	10_{134}	23	3
8_5	24^{1}	8	9_{34}	21	1	10_{35}	30	23	10_{85}	26	2	10_{135}	23	3
8_6	23^{1}	8	9_{35}	24	4	10_{36}	26	6	10_{86}	24	1	10_{136}	21	3
8_7	23^{1}	1	9_{36}	25	2	10_{37}	28	48	10_{87}	24	2	10_{137}	22	2
8_8	24^{1}	12	9_{37}	24	9	10_{38}	27	6	10_{88}	22	9	10_{138}	26	4

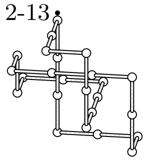
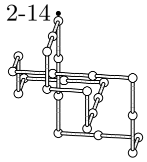
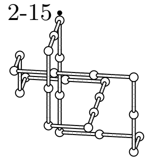
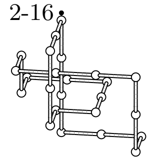
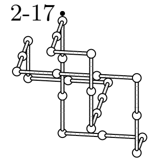
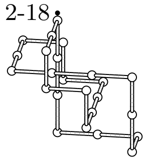
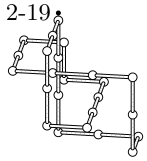
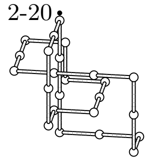
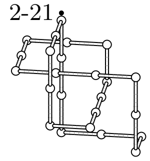
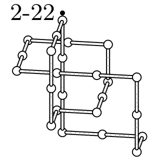
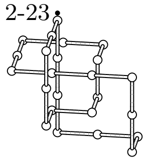
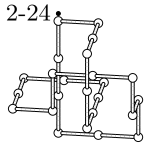
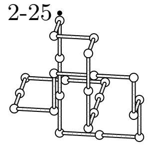
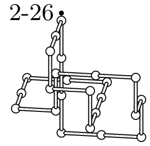
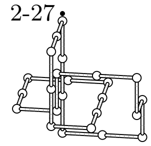
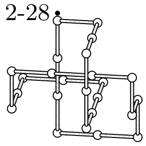
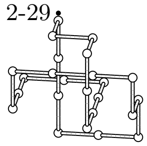
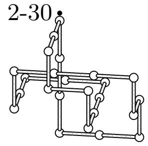
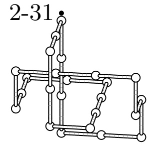
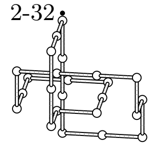
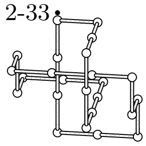
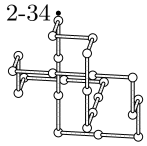
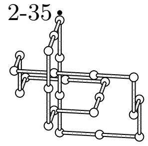
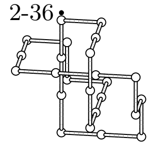
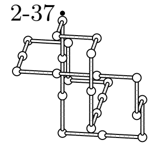
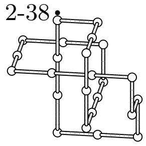
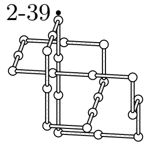
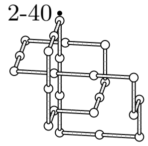
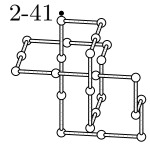
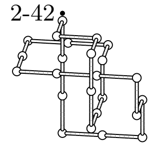
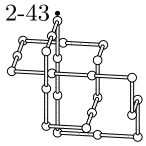
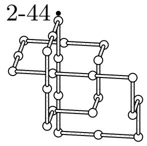
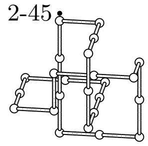
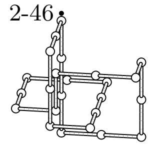
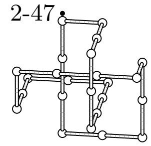
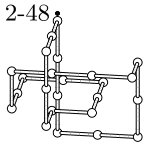
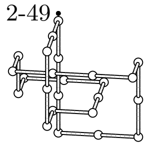
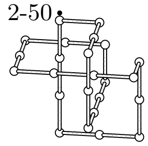
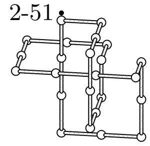
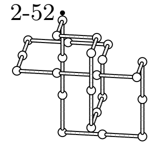
K	S	C	K	S	C	K	S	C	K	S	C	K	S	C
8_{9}	22[1]	6	9_{38}	25	1	10_{39}	26	11	10_{89}	25	4	10_{139}	20	1
8_{10}	22	4	9_{39}	24	9	10_{40}	26	6	10_{90}	25	9	10_{140}	21	1
8_{11}	25[1]	2	9_{40}	18	1	10_{41}	26	19	10_{91}	25	9	10_{141}	21	1
8_{12}	26[2]	7	9_{41}	21	1	10_{42}	26	4	10_{92}	28	10	10_{142}	24	4
8_{13}	21[1]	25	9_{42}	20	4	10_{43}	24	9	10_{93}	27	23	10_{143}	23	8
8_{14}	25[3]	4	9_{43}	21	32	10_{44}	23	9	10_{94}	25	8	10_{144}	23	2
8_{15}	22[1]	18	9_{44}	21	41	10_{45}	23	1	10_{95}	28	4	10_{145}	22	3
8_{16}	23	1	9_{45}	21	14	10_{46}	28	19	10_{96}	27	16	10_{146}	22	13
8_{17}	20	6	9_{46}	20	5	10_{47}	28	4	10_{97}	30	1	10_{147}	24	8
8_{18}	20	7	9_{47}	18	6	10_{48}	28	4	10_{98}	29	4	10_{148}	24	10
8_{19}	20[1]	24	9_{48}	23	24	10_{49}	26	30	10_{99}	28	2	10_{149}	22	1
8_{20}	19[1]	2	9_{49}	20	2	10_{50}	27	2	10_{100}	27	6	10_{150}	22	8
8_{21}	20[1]	18	10_{1}	29	87	10_{51}	28	16	10_{101}	27	3	10_{151}	23	2
9_{1}	27	24	10_{2}	27	39	10_{52}	26	23	10_{102}	25	8	10_{152}	24	3
9_{2}	26	15	10_{3}	28	2	10_{53}	26	7	10_{103}	25	16	10_{153}	24	33
9_{3}	26	10	10_{4}	27	17	10_{54}	28	4	10_{104}	25	1	10_{154}	25	60
9_{4}	26	2	10_{5}	27	4	10_{55}	28	29	10_{105}	28	9	10_{155}	23	2
9_{5}	24	4	10_{6}	29	4	10_{56}	27	4	10_{106}	25	2	10_{156}	22	11
9_{6}	26	10	10_{7}	27	18	10_{57}	28	13	10_{107}	26	1	10_{157}	24	2
9_{7}	25	12	10_{8}	27	60	10_{58}	28	3	10_{108}	25	2	10_{158}	25	3
9_{8}	23	18	10_{9}	26	28	10_{59}	26	2	10_{109}	26	4	10_{159}	23	34
9_{9}	25	11	10_{10}	25	6	10_{60}	26	10	10_{110}	30	2	10_{160}	22	8
9_{10}	24	4	10_{11}	27	1	10_{61}	28	16	10_{111}	26	8	10_{161}	23	2
9_{11}	26	23	10_{12}	25	2	10_{62}	26	4	10_{112}	24	4	10_{162}	24	4
9_{12}	25	6	10_{13}	30	13	10_{63}	26	12	10_{113}	24	2	10_{163}	21	6
9_{13}	23	1	10_{14}	28	2	10_{64}	26	11	10_{114}	23	2	10_{164}	20	1
9_{14}	23	6	10_{15}	28	18	10_{65}	25	4	10_{115}	27	2	10_{165}	23	4

Table 6: The complete list of 75 canonical lattice polygon of step number 24 that are trefoil knots. Their geometric realizations are provided after the list.

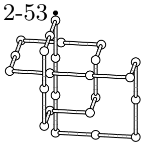
1-1	DDDEENUUWSSDWWNNEESEUUWW	1-2	DDDEENUUWSSWDWNNEESEUUWW
1-3	DDDEENUUWSSWWDNNEESEUUWW	1-4	DDDEENUUWSSWWNDNEESEUUWW
1-5	DDDEENUUWSSWWNNDEESEUUWW	1-6	DDDEENUUWSSWWNNEDESEUUWW
1-7	DDDEEUUWSWDWNNEESSEUUWNW	1-8	DDDEEUUWSWDWNNEESSEUUWWN
1-9	DDDEEUUWSWWDNNEESSEUUWNW	1-10	DDDEEUUWSWWDNNEESSEUUWWN
1-11	DDDEEUUWSWWNDNEESSEUUWNW	1-12	DDDEEUUWSWWNDNEESSEUUWWN
1-13	DDDEEUUWSWWNNDEESSEUUWNW	1-14	DDDEEUUWSWWNNDEESSEUUWWN
1-15	DDDEEUUWSWWNNEDESSEUUWNW	1-16	DDDEEUUWSWWNNEDESSEUUWWN
1-17	DDDEEUUWSWWNNEEDSSEUUWNW	1-18	DDDEEUUWSWWNNEEDSSEUUWWN
2-1	DDDEESUUWWDWNNEESSSUUNNW	2-2	DDDEESUUWWDWNNEESSSUUNWN
2-3	DDDEESUUWWDWNNEESSSUUWNN	2-4	DDDEESUUWWDWNNEESSSUWUNN
2-5	DDDEESUUWWDWNNEESSSWUUNN	2-6	DDDEESUUWWWDNNEESSSUUNNW
2-7	DDDEESUUWWWDNNEESSSUUNWN	2-8	DDDEESUUWWWDNNEESSSUUWNN
2-9	DDDEESUUWWWDNNEESSSUWUNN	2-10	DDDEESUUWWWDNNEESSSWUUNN
2-11	DDDEESUUWWWDNNEESSWSUUNN	2-12	DDDEESUUWWWNDNEESSSUUNWN
2-13	DDDEESUUWWWNDNEESSSUUWNN	2-14	DDDEESUUWWWNDNEESSSUWUNN
2-15	DDDEESUUWWWNDNEESSSWUUNN	2-16	DDDEESUUWWWNDNEESSWSUUNN
2-17	DDDEESUUWWWNNDEESSSUUWNN	2-18	DDDEESUUWWWNNEDESSSUWUNN
2-19	DDDEESUUWWWNNEDESSSWUUNN	2-20	DDDEESUUWWWNNEDESSWSUUNN
2-21	DDDEESUUWWWNNEEDSSSWUUNN	2-22	DDDEESUUWWWNNEEDSSWSUUNN
2-23	DDDEESUUWWWNNEESDSWSUUNN	2-24	DDDEEUSUWWDWNNEESSSUUNNW
2-25	DDDEEUSUWWDWNNEESSSUUNWN	2-26	DDDEEUSUWWDWNNEESSSUWUNN
2-27	DDDEEUSUWWDWNNEESSSWUUNN	2-28	DDDEEUSUWWWDNNEESSSUUNNW
2-29	DDDEEUSUWWWDNNEESSSUUNWN	2-30	DDDEEUSUWWWDNNEESSSUWUNN

2-31	DDDEEUSUWWWDNNEESSSWUUNN	2-32	DDDEEUSUWWWDNNEESSWSUUNN
2-33	DDDEEUSUWWWNDNEESSSUUNNW	2-34	DDDEEUSUWWWNDNEESSSUUNWN
2-35	DDDEEUSUWWWNDNEESSWSUUNN	2-36	DDDEEUSUWWWNNEDESSSUUNNW
2-37	DDDEEUSUWWWNNEDESSSUUNWN	2-38	DDDEEUSUWWWNNEEDSSSUUNNW
2-39	DDDEEUSUWWWNNEEDSSSWUUNN	2-40	DDDEEUSUWWWNNEEDSSWSUUNN
2-41	DDDEEUSUWWWNNEESDSSUUNNW	2-42	DDDEEUSUWWWNNEESDSSUUNWN
2-43	DDDEEUSUWWWNNEESDSSWUUNN	2-44	DDDEEUSUWWWNNEESDSWSUUNN
2-45	DDDEEUUSWWDWNNEESSSUUNNW	2-46	DDDEEUUSWWDWNNEESSSWUUNN
2-47	DDDEEUUSWWWDNNEESSSUUNNW	2-48	DDDEEUUSWWWDNNEESSWSUUNN
2-49	DDDEEUUSWWWNDNEESSWSUUNN	2-50	DDDEEUUSWWWNNEEDSSSUUNNW
2-51	DDDEEUUSWWWNNEESDSSUUNNW	2-52	DDDEEUUSWWWNNEESDSSUUNWN
2-53	DDDEEUUSWWWNNEESDSWSUUNN	2-54	DDDEEUUWSWWDNNEESSWSUUNN
2-55	DDDEEUUWSWWNDNEESSWSUUNN	2-56	DDDEEUUWSWWNNDEESSWSUUNN
3-1	DDDEENUUWSSDDDNNUUSEUUWW		

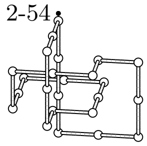


2-13

2-14

2-15

2-16

2-17

2-18

2-19

2-20

2-21

2-22

2-23

2-24

2-25

2-26

2-27

2-28

2-29

2-30

2-31

2-32

2-33

2-34

2-35

2-36

2-37

2-38

2-39

2-40

2-41

2-42

2-43

2-44

2-45

2-46

2-47

2-48

2-49

2-50

2-51

2-52


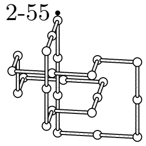
2-53



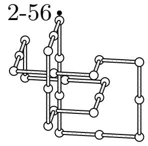
2-54



2-55



2-56



3-1

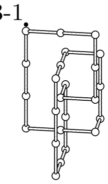


Table 7: Examples of potential minimum step number lattice polygons representing prime knots up to 10 crossings.

K	NEWSUD Form
0_1	DEUW
3_1	DDEEUUWSWWNNEDESSEUUWWN
4_1	DDDDEEUUWUNNDDSSSUWWNNEEEUUWWS
5_1	DDDEEUUSSUUNWNDDSWWNNEDESSEEUUWWWN
5_2	DDDEEESSWWNDNNUUUSSEDDDSWWNUUENEUUWW
6_1	DDDEEENNWWUSSSWWNNUEESSEENDNWDDSSUUNUUWW
6_2	DDDEDNNUUSSSWDWNNNUUSEEEDDWWDWWUUEENUUSS
6_3	DDDDEENUUUWWWSDSEENNNNDDSSUSEENNWWWUUUSS
7_1	DDDEESSWWSUUNNDDWWUUENNEESDSDDSSUUNWWNNWUUEE
7_2	DDDEEUUNUUSSDWWDWNNEESSSUUNNDEEDNDWWSSWSWUUNNE
7_3	DDDEENUUUWWDDDWWUUESSEENNNEESSWDWNNWWSSWUUEE
7_4	DDDDDEEUUWSSUUNNNEDDWWWSSUEEEDDWNNNUUSSEUUWW
7_5	DDDDEEUUUSWWWNNEENDDSSSEENUNWWWNUUSESDDSSUUNWN
7_6	DDDDDEEUUUUWSSDDNNUNNDDSSSEENNUWWWWSSEUEEUUWNW
7_7	DDDDEEENUNWWSSSSUUNNNEDDDDSSUUUWWWNNEESEUUWW
8_1	DDDDEEUUEESSWWWNNNWWUSSEEEDDSESUUNNDDNNWWSSSWUUUNN
8_2	DDDDEDNNUUSSSWWNNENNUUSSESSDWWNNEEEDDWWWNUUEEUUSWS
8_3	DDDEEUUENNWWSSSWWNNWDDEEUUUEEDDESSWWNNNWWSSWUUEE
8_4	DDDDEEDNNUUSSUUSWWWNDNEEEEDDWWDSSUUNNNUUSSDSWSUUNN
8_5	DDDDEEUNNNWWSSWWUUEEENDDDWWUUSSDSEENNDEEUUWWWNUUSS
8_6	DDDDEEUUSUWWWNNEEENDNWWSSEEDESSWWNNNUUUSSDDSWSUUNN
8_7	DDDDEEDSSUUNNNUUWSSSDDNNNDDSSUEEUUWWWWNNEESEUUWW
8_8	DDDDEEUUWSWWNNEEENNUWWSSEESESSWWNNUNNDDDSSSEUUUWWN
8_9	DDDDEEEUNUWWWWSSEENEESSWUWNNNNDDSSSDEDNNUUSSUUNUWW
8_{10}	DDDDEEUUUSWWWNNEEENDNWWSSEEESSWWNUUNNDDDSSSWUSUUNN
8_{11}	DDDDDEEUUWNNEESSSWWSUUNNNEDDDWSSESUUNWWWNNEESEUUWW
8_{12}	DDDDEEUNUUWWWNNEESDSSEENNWDWWSUWWDDEEEUSSWWNNNUEUUSS
8_{13}	DDDDDEESUUUUWWWNNDEESSSSDDNNNNWWUSSEENEESSWWWUUUNN
8_{14}	DDDDDEEUUSUWWWNNEEENDDSWSSEENUNWWNUUSSSSDDNNUEUUWW
8_{15}	DDDDDEEUUUWNNNDDSSSSEENNNWUWWWUUSEEEDDWSSWWNUNENUUSS
8_{16}	DDDDEESUUENNWWWWSSEESSDDNNUNNNUUSSSSWDDEEENNWUUUWW
8_{17}	DDDDEEEUUNWWWWSSEEEDDWWSUUUNENNNDDSSSSDDNNNEUUSUUUWW
8_{18}	DDDDDEEUUUNUWWWDWDDEEESSSUUNNNNDWWSSSEEEENNWWWUNUUSS
8_{19}	DDDDEESUUUWNNWDDWSSSEUENNNNWWSSSEDENEUUUWW
8_{20}	DDDDDEEUUUWSWWNNEEEDDWSSSUUUNNNNDDSSSEUUUWNW
8_{21}	DDDDEESUUUWWWNNEEEDDSWSWUWSSEENNNNDWWSSSEUUUNN
9_1	DDEEDDNNDWWUUNNWWSSEUUSEESSDDNNEEDDWWUNNWWUUWSSEENUUSS
9_2	DDDEENEESSEUUWNWDDDNWWUUSSWWUNNEESSEEDDDEENUUWSSWWNNUUWW
9_3	DDDEEEDDSWWWUUEEEUUWWNNWWSSDEENNDDSSSDDENNUUSSUUNNUUWW
9_4	DDDEEUEESSWWNUUWWDDDWWUUENNEESSSEENDDNNWUWSSSWWNNWUUEE
9_5	DDDDDEEEUUWWSSUUNNNEDDWWUWSSEEEDDNNDWDSSUUNNNUUSSEUUWW
9_6	DDDDEEEUUWWSSUUNNDNNDDSSSDDNENUUSSUUWWWNNEUEEDDWWWWUSUEE
9_7	DDDEEDDWWWDDEESUUWWUUEEEUUWWSDDDNNNDDSSSEUUNNNUUSSSUUNNW
9_8	DDDDDEEUUNUUWSSSDDEDDWWUUNEEDENNWWSSSSEUUNNWWWNNEESEUUWW
9_9	DDDDEEDNNUUSSUSUWWWNNEENDDSSSDDNNUEEUUWWWNUUSESDDSSWUUNN
9_{10}	DDDDEEUUSSEENNWNWWWNNEESUSWWSSEEEEDDWWNNNUUUSSDDSWSUUNN

K	**NEWSUD Form**
9_{11}	DDDDEEUUESSWWNNNWWSSSDEENNNEESSWUUWWUSSDDNNDWWUUESEEUNNW
9_{12}	DDDDEEDSSUUNNNUUWSSWWWDDEEUEEEDDWWDNNUUSSSWWUNNNEESEUUWW
9_{13}	DDDDEENUUWWDDWWUUENNEEEUSSWWWNWDDESSSEENNNUUNNDDSWSUUUSS
9_{14}	DDDDEEEUUWWSWWNNEEEEUUSSWDDDNNDDWSSUUNNNUUSSSSEENNWUWW
9_{15}	DDDDEDSSUUNNNWWNUUSSSWWNNEDEENDDWSSUUSSDDNEEEUUWWDWWUNUUEE
9_{16}	DDDDDEEUUWNNUUSSDSSDDNNNEUUWWWSSESSWWNNUEEEEDDWWSWUUUNEN
9_{17}	DDDDDEEUUUUNWWWNDDESDESSEUENNWWSSSUUNNDNEESSWWWWNNENUUSS
9_{18}	DDDDEEUUNEESSWWWWSUUEENNEDDDNWWSSSUUNWWNNEESUUSEEDDWNNUWWS
9_{19}	DDDDDEEUUSUWWWNNEEENNDWWSSEEDESSWWNNNUUUSSSSDDNNUEUUWW
9_{20}	DDDDDEEUUWNNUUSSSSWDDDENNNEUUWWWSSSSEDENNWWWUUEEDEEUUWWN
9_{21}	DDDDDEEUUUUNWWWNDDSSSEEESUUNWNDNEESSWWSDDNNNWWWUUEENUUSS
9_{22}	DDDDEEEENNWWWWUSSSEENNNNDEDSSUUUUWWWSSEDENNWWWDDEEUNUUUSS
9_{23}	DDDDEENUUUUNNDDWSSSSWWNNNDDEESUSEENNWWWSWWUUEEESEENNWWUSSW
9_{24}	DDDDDEEUUWUNNDDSSSSUUUWNWWDDEEEESSWWUNNEENNDWWWSUSSUUNEN
9_{25}	DDDDDEEUUWSWWNNNUUSEEEDDWWDDNNUUSUSWWNDDEEESSSSUUNNNNUUSSW
9_{26}	DDDDEEEEESSWUWNNENNWWSSSSUUNNNEDDDDSESUUUWWWWNNEESEUUWW
9_{27}	DDDDEEEEENNWWWUUUSSSSDDNNNWNNEESSSSDEDNNUUUWWWWSSEEEUNUWW
9_{28}	DDDDDEEUUNUUWSSSDDDNNNEESSUWWWWWUUEEDEESSWWNWNNNEESEUUWW
9_{29}	DDDDEESUUUWNNUEEDDDSSSWWNNNNUEEESSWWWSWWNNEEEDNNUUSSSUWW
9_{30}	DDDDDEEUUWNNWWUSSSEDEEEUNNNWWSSSSDDNNNEUUUWWWWDDEEUNUUSS
9_{31}	DDDDDDEEUUWSWWNNEEEUUUWSSSDDNNNNDDSSSEUUUWWWWNNEESEUUWW
9_{32}	DDDDEEEENNWWUSSSSUUNNNNDEESSSWWWWNNEEEUNNDDSSDDSSUUNUUUWW
9_{33}	DDDDDEEUNUWWWSSSWUUNEEEEDDWWDWWUUESSEENNNNNDDSSSSWUUUUNN
9_{34}	DDDDEEEEUUWNNDWWSSSSUUNNNEEEDDWSSUWWWWNNEEEDDDSSUUNUUUWW
9_{35}	DDDDEENUUUESEDDWWUUSWWWNNEENNDDSSSSWWNNUEESSEENNNWWWUUSS
9_{36}	DDDEEEESSWUUNUEEDDWWDDNNUUUESSSUWWDNNNWWDSSEENNNEESSWUUWW
9_{37}	DDDDDEEUUUUNWWWWDDEEESSEENNWDWSSSUUUNNDNEESSWWWWNNNUUSSE
9_{38}	DDDDEENUUESSWWSUUNNDDNNNDDSSUSSSEUUNWWWNNEEESDDENNWWWUUUSS
9_{39}	DDDDEEEESSUWUUWNNNDEESSSSDWWNNNNWWSUSEEEUEEDDWWDDNNUUSUUUWW
9_{40}	DDDDDEEUUUSUWWWWDDEEEEENNWWWWUSSSSEENNNNNDDSSSSUWUUUNN
9_{41}	DDDDEEUNNNUUSSDWWWSSEEEUENNDDDWWUSSSUUNNNNDEEESSWWUUWW
9_{42}	DDDDEDNNUUSSSWUUWWNDNEEESSSWWNNUNEEEDDDWWNUUUUSS
9_{43}	DDDDEEUUSUWWWSSEEDNNNNWDDWWSSUEEENNNWWSSDSSUEUUUNN
9_{44}	DDDDEESSUUNNUWUNNDDSSSSUEUNNNDWWWSSEEEEDDWWWSUUUNN
9_{45}	DDDDEEESUUWWWWNNDEESSSUEUUENNWDDDSSSWUUNNNNEESSWUWWN
9_{46}	DDDDEEDSSUUNNUWUSWWSSDEDENNNNWWSSUEEEEDDWWSWUUUUNN
9_{47}	DDDDDEEUUUUWNWDNDEDSSSSUUNNNNEESSSDWWWWNNEEEUUUWWS
9_{48}	DDDDEDNNUUSSSWUWUNNNDDSSSSUEUNENNEDDDWWWWUUEEESEUUWW
9_{49}	DDDDEESUUUWWDWWWNNEESDSEENEUUWNWWSSSEDENNNWDWWSUUUEE
10_{1}	DDDEEDSSUUNNNEESSDWWWDNNUUSSWWNUNEESSUUENNDDSSEEEDDNWWUUUWWW
10_{2}	DDDDEESUUWWWNNEESEEUUUNWWWDDESSSDDNNENUUSSEENNWUWSUUNNDDWSSW
10_{3}	DDDEEEDDWWDSSUUNNNUUSSWWNNUEESSEEDDWWWDDEEDNNUUSSSUUNNUUWW
10_{4}	DDDDEEEUUWWDSSUUNNNWNNNEESSDDSSSUWWWNNEEEENNWWUUWWSDDEEUUWSS
10_{5}	DDDDEEDDSSUUUNNNNUUWSSSDDNNNDDSSEESSWWNUNEEUUWWWWNNEESEUUWW
10_{6}	DDDDEEUUUSWWDDWWUUENNEEENNWWSDSEEESSDWWNNNUUUSSDDSSWWNNWUUEE
10_{7}	DDDEEDDNNUUUWSSSWWUNNEESSEUUNNDDSDSDWDNNEEEUUWWDWNNEESSSUUUWWW
10_{8}	DDDDEEDNNUUSUUWWWSSWWNNEDDEEESSDDNNUEEUUNWWSSSWWNNNWWSSEUUEE
10_{9}	DDDDEEUSUWWDWWUUEEEEESSWWSSEENDNWWWNNNDNEESSSUUUSSDDWDNNUUUNN
10_{10}	DDDDEEUEUSSDDNNENNWWSUUUWSSSDDNNNNEESDDESSUUWWUWWWNNEESEUUWW
10_{11}	DDDDEEDNNUUSSUSUWWWNNEEENNWDWSSEEEDDWWDSSUUNNNUUUSSDDSSUUWNN
10_{12}	DDDDEEEDDWWSUUEEUUWWWWDNNEESSSUUNNNEDDDDSSSUUNNENNWUWSSEUUWW
10_{13}	DDDDDEEUUESSWWNNNWWSSSWUUNEEESSDWWNNWDDEEUUEEDDENNWWSSSWUUUUNN
10_{14}	DDDDDEEDNNUUSSSUUWWWNNEENDDSSSDDNNUEEUUWWWNUUESSSSDDNNEUUUWW
10_{15}	DDDDDEEUUESSWWNNNWWSSSDEENNNEESSWUUUWWWSSEEDNNWWWDDEEUSUUUNN
10_{16}	DDDDDEEEUUWWNWWSSEEESUUNWNNEDDDSSDDWNNUUSSSUUNEENNNWWSSEUUWW
10_{17}	DDDDEEDNNUUSSSUUNNUWUSSDDNNNDDSSSDDNNUEEUUWWWWSSEENEEUUWWW
10_{18}	DDDDEEDNNUUSSSUUNNNNWWSSEUSEEDDWWSWWNNEEEEDDWWDSSUUNNNUUUWSS

K	**NEWSUD Form**
10_{19}	DDDDEEEUUWWSWWNNEEEEUUWWDWWUNNEESDDDSSSDDNNEUUSSUUNNNNWWSUSS
10_{20}	DDDDEESUUWWDDWWDNNUUSSUUEEESSWWNDNNWWDDEEDSSUUNNNEESSSUWUUNN
10_{21}	DDDDEEEDDNWWWUUEEEUUWWWWSSEUENNNDDSSSDDNNNDDSESUUNNUUSSUUUWW
10_{22}	DDDDEEEUUWWUUEEDSDWWWWNNEEENNWWSUSEESSSDDNNNDDSSUWUNNNUUUSWS
10_{23}	DDDDDEEUUENEDDWWUUUUWSSSDDNNNWDWSSEENNNEESSSUUWWWWNNEESEUUWW
10_{24}	DDDDEEEUUUNNDWDSSSDWDNNUUSSSEUUNWWWNNEEEDEESSWWWSUUNNDDNNUUSWS
10_{25}	DDDDDDEEUUWSSEENNNWWWNNEEUSSWSSSUUNNNNDDDDSSSEUUUWWWNNEEEUUWSW
10_{26}	DDDDEEUUUEEDDDWSWUUUWWDWNNEEEDESSEENNUWWSSSWDWNNNNUUSSDSSUUNNW
10_{27}	DDDDEEEENNWWSUSEESSWWNUUWNNNDDSSSSUEENDDDNNUUUWWWWSSEENEUUWW
10_{28}	DDDDDEEUUWSSUUUUNNDDSWSSEENNNNDDWWWSUSEEEDDWNNNUUSSEEUUSWWWN
10_{29}	DDDDDEDNNUUSSSWWNNEUESEEUNNWWSSSWWNDNNUUSEEEDDDDWWWWUUUEENUUSS
10_{30}	DDDDDEEUUESEDDWWUUUWWWNNEEENDDWSSSSEENNNWUWNUUSSSSDDNNUEUUWW
10_{31}	DDDDDEEUUUUNWWWNDDSSSEEESUSWWNNEEENNWWSUSSDDDNNNWWWUUEENUUSS
10_{32}	DDDDDEEUNNUUUWSSDSSDDNNNDEEUUWWWWSSEEEUENNNNWWSSEUEESDDWWUUUWW
10_{33}	DDDDDEEUNUUSSSSWWNNENNUWWSSEEEEDDWWWWNNEESEEDNDWWUSSSUUUUNNW
10_{34}	DDDDEEUUWSWWNNEEEDDWWUNNUUSSEESESSWWNNUNNDDWWDDEEUSSSEUUUWWN
10_{35}	DDDDEEDNNUUSSSUUENNDDDWWDSSUUNNNNUEUSSDWWUWSSEENNNDEEESSWWUUWW
10_{36}	DDDDEENEDDWWUUUSSSUUNNNEDDSSSDDNNNNWWSSSUEEEUUWWWWNNEESEUUWW
10_{37}	DDDDEEUNNNWWSSUEEEDDWWUSSWWNNNNWWSUSEENNEUESSWWWNDDDEEEUUWUUSS
10_{38}	DDDDDEEUNNWUUSSDEENDDWWUSSSUUUNNNEEDSSWWWSSDEENNWWWNNEEEUSUUWW
10_{39}	DDDDDEESUUWWWNNEEENNWUWSSEEEUSSSWDDWDNNNNUUUSSSEDESSWWNNNEUUWW
10_{40}	DDDDDEEUUSSUUENNDDSDWWNNENUUSWWWSSEEEEEDNNWWWNUUSSSSDDNNUEUUWW
10_{41}	DDDDDEEUUUUWNWWNDDDDESEUSSWWNNENNUWWSSEEESSEENUNWWSSWWNNENUUSS
10_{42}	DDDDDEESUUWWWNWWSSEEEEUUNWWWDDDDEEEUNNWWSSSSUUNNNNEESSWSUUNN
10_{43}	DDDDDDEEUUUNUUWSSSDDDDNNNEEUSSWWWWNUNEESSWSSEEUNNWWWNNEESEUUWW
10_{44}	DDDDDEEEUUWWSWWNNEEEEUUWWWWSSEENUNNDDSSSSEENNWNNDDSSSUUUUWWN
10_{45}	DDDDDEEEEUUWWNDDDSSUUUUWWWNNEEEEDDSSSWWNNNUNUUSSSSDDNNUEUUWW
10_{46}	DDDEEEDSSUUUNNNWDDDSSUEESSWUWNNEEUUWWDWNNWDWSSEENNNEESSWUUWW
10_{47}	DDDDEEESUUWWWWNNEEEDSSSUUNNNEESSWUUWWDDNNNDDSSUEEUUESSWWNNWW
10_{48}	DDDDEEEUUWWUSSDDNNNDDESSUUNNNUUNWWSSEESSDWWWNNEEEEENNWWSUUWSS
10_{49}	DDDDEEEUUWWSWWNNEEEUUWWDNDDESSSDEDNNUUSSUUUENNDNWWSSSSEENNWUWW
10_{50}	DDDDEEEUUUSSDDWWNNNDDSSUEUNNNUUSWWWSSEEEDEENNWWWNUUSSDDSSUUNWN
10_{51}	DDDDEEUUSUWWWSWDDWWUUNEEDDDEEUUESSWWNNNWWSSSEDENNNNEESSSWUUUNN
10_{52}	DDDDDEEESUUUSWWNNNNDDSSUEEESSWWNUUWWWSDDNNNEEEDSSWWWWUUEESUUNN
10_{53}	DDDDDEEUUNUUWSSSDDSEENNNNDWWSSSEUUSWWNNEEEEDDWWUWWWNNEESUEUUWW
10_{54}	DDDEEEDDWSWUUEEEUUUWWDDSDDNNNUUUESSSWWNNNWWSSDEENNNEESSWUUWW
10_{55}	DDDDDEEUUUWNNNDDDSSUSSEENNNWWWNNEUESSWWWUUSEEEDDWSSWWNNUENUUSS
10_{56}	DDDDDEEEUUWWUSSDDNNNDDSSEUUNNNUUSSESSWUWNNNEEDDWWWWSSEEEUUUWWN
10_{57}	DDDDDEESUUWWWNNNDEESSSSUUUUNNNDWWSSEEESSWWNDNENNWWWDDEEUNUUUSS
10_{58}	DDDEENNWWWUSSSEUENNWWWDDEEUNNNEESSWUUUESSDDNNNWDDDSSUUEEUUWWSW
10_{59}	DDDDEEUEEEUUWWDNDDSSUUUUWWDNNEEEDSSSWWWDNNNUWWSSEEEDSSUUNNNUWW
10_{60}	DDDDDEEUUWUUSSDDNDNNNUUSWWSSEEENNNUNUWWSDDDSEEEUUWWWWNNEESUSSW
10_{61}	DDDEEDDSWWUUEEEUUWWNNWWSSESDDSESUUNNDDDNNNUUSSWWSSEEENNNUUWW
10_{62}	DDDDEEEUUWWUUSSDDDNNNDEDSSUUNNNUUSSSWWSDSEENNWWWNNEEEEUUWWWS
10_{63}	DDDDDEEUSUWWWNUNNUEESSSEENNWDDWWWWUUEEDNDDESSSSUUNNNNEESSWUUWW
10_{64}	DDDDEEUNNEESSSDWWUUUWWWNNEEDESEEEDDWWUSSWWNNNDEEUUUWSWDSWSUUNN
10_{65}	DDDDDEEUUUUNWWWSWWNNDEESSSEEEENNWWSUSSDDDNNNEEUSSWWNNWWWUUUESE
10_{66}	DDDDDEEUUENNWWSWWSWSUUENNNEDDDESSSEENNWUUUWWDWNNEESSSSWWNWUUEE
10_{67}	DDDDEEUENNWWSSSUSEEENNWWWNNUUSSDSWWNNEEEDDDESSUUUSSDDNWNUUUWWN
10_{68}	DDDDDEEUUNWWWSSSWUWNNEESSSEENDDNNNNUUUSSSWWWDDDEEUEESSWWNUUUNN
10_{69}	DDDDDEEUUUUNWWWNDDDSSSEENNDWWWUUEEESSSUUNNDNEESSWWWWNNENUUSS
10_{70}	DDDDDEEEUNNWWSSSSUUNNENDDDEEUSUWWWNWWSSSUUNEEEDDWWDWWUUEESUUNN
10_{71}	DDDDEEUEENUUWWWSSEDDDDEEUUWUSSDDNNNNUUSESSDWWWNNNWUWSSEENEUUWW
10_{72}	DDDDDEEUNNNWWSSWSSEENNDEEUUWWWWWUUEEENDDDDWWUUUSSSEENNWNUUSS
10_{73}	DDDDDEENUUWWWSSSEEEENNWWSDSSUUNNNNNDDSSESUUUWWWWDDDEEUUSUUNN
10_{74}	DDDDDEEUUWUNNDDSSSSUUUWNWNWWSDSEENNDNEEEUSSWWSSDEENNWWWWUUUEEN
10_{75}	DDDDEEENUUUNWWSWWWDDEEESSSEENNWNDDSSUUUUNNDNEESSWWWSWWNNNUUESS

K	**NEWSUD Form**
10_{76}	DDDDEEUUUNWWWSSEEEESSWWWDNNEEENNDWWSSSEUUSSDWDNNUUNNDDNNUUSWS
10_{77}	DDDDENNUUSSSWWNWWDDEEUNNWUUSSSDDNNDEEUEEUUWWWSUUNNDDDSWWUUEEES
10_{78}	DDDDDEEUNUUUWSSSDDNNEEEDDWNWWUSSSEENNNEDDSWSWUUUUWWWNNEESEUUWW
10_{79}	DDDDEEUUWNNWUWSSSEEESDSWWNNEEEUNNNWWSSUSSDDDNNNEUUUWWDDWSWUUEE
10_{80}	DDDDDEEUUWNWWWUUEEEEDDDWSSSUUUNNDNNEEDSSSWWWSSEENUNWWWNNENUUSS
10_{81}	DDDDDDENEEUUWWWWSSEDENNNUUSSSSUUNNDEDDDNNUUUUSSSDWWWNNEEEEUUWWSW
10_{82}	DDDDEEDNNUUSSSSUUWNNNNDDSSSDDNNUEEUUUSSDWWWWNNEEENEESSWWUUWW
10_{83}	DDDDDEEEUUWWDSSUUNNNNNWWSSSWDDENEEUEUUWWWWDDESSEEEDNNNWUWUUUSS
10_{84}	DDDDDEEUUUSSUUNWNDNWWSSEEEEDDWWWWNNNDEESSSSUUNNNWWWDDEEUNUUUSS
10_{85}	DDDDEESSEUENNWWSSWUUNNNNDDSSSDDEEUUUUNNDWWWWSSEEESEENNWWUUWW
10_{86}	DDDDDDEEUUUUWNNDDDSSSSUUNNNEUUWWWNDDSSDSEEEENNWWWWWUUEENUUSS
10_{87}	DDDDDEEUUEENNWWSWWWSSEEEEDNNUUNUWWDDDSSSSUUUNNNEDNNWWSSESEUUWW
10_{88}	DDDDDEEUNUUUWWWWDDEEENEESSWWDNNNNUUSSSSSWWNNNNDEEEDSDWWUUUUUSS
10_{89}	DDDDDEENUUWWWWUUEEEEDNDWNNUUSSDSSSWWNNNNEEEDDWSSSSWWNNNEUUUUSS
10_{90}	DDDDDEEEUUUSSDWWNWWNNEEEEESSWWUUWWDSDDEENNNDDSSWUUNNNUUSSSSUUNWN
10_{91}	DDDDDEEUUUEEDDNWWUUUWWWSSDEDEEENNNWWSWWNNEUESSSSSDDNNNNWUUUUSS
10_{92}	DDDDDEEUUWNNNWWSSSSEEEUUWNNNWDDSDWWUUEEENEESSWWSSDDNNNEUUUUWSW
10_{93}	DDDDEEEUUNWUUUWWDDESSEDDNNDWDSSUUNNWNUUEEEESSWDWSWWNNEENNUUSSSW
10_{94}	DDDDDEEUNNUUESSSWWWSSEDENNWWWNNEEEEEUUWWWSSSDDDNNNDEEUSUWUUUWW
10_{95}	DDDDEEUUUESSWDWWSWWNNEEENNEESSWUUWWDWNNEESSSSDDNNNNWWSSSUUUNNE
10_{96}	DDDDDEEUUUNUWWWWNNDEESSSUSSDDNEENNWWWWUWWDDEEESSESUUNWWWNDNNEUUUSS
10_{97}	DDDDEEEUUWWNWWSSESUSEENNNNDDSSSWWWUUNEENUUSSDDDSDDNNUUEEUUWWWN
10_{98}	DDDDEEESUUWWWWNNEENNDDSSUSSDDNENUUUUSSSDDNNENNWWWUUESSDEEUUWWW
10_{99}	DDDDDEEUUWNWNUUEEEDSSWWSSDDNNNEUUUUUWWDDESESDDWWWUNNEENNUUSSSW
10_{100}	DDDDEEENUUUNWWWDDSESSDEDNNUUSSUUWNNUNNDDSSWWSSEENEEENNWWSUUSWW
10_{101}	DDDDDEESUUENNWWWWSSEESSDDNNUNNNUUSSSSSWUWNNEENEDDDESSWWWUUUUNN
10_{102}	DDDDEDNNUUNWWSSSSEENNEEUUWWWWNNEDESSWWWDDEEEESUUNNNDDWWSUUUUSS
10_{103}	DDDDDEESUUWWWWUUEEENNEDDWWWSUSSSEEENNNWNNDDSSSSUUUSWSDDNNUUUNN
10_{104}	DDDDEEESSWWWWUUNNNEEEDDDSSUUUUWNNNDDSSSDDSSUUWNNWWUUEEDENEUUWW
10_{105}	DDDDEEESSWWNUNNDNNUUSSWWSSEEEENNWDDDSSUUSUUWNNNWNDDEEESSWUUUWW
10_{106}	DDDDDEEUUUEESDDWWUUUWWSDWDNNNEEEESSSWWWNUWWDDEEENNNUUSSSSUUNNW
10_{107}	DDDDEEEUUWWSWWNNNNEESSEEUUWWDWWWDDEEENNEESSWSSUUNNNDDDWWUUUUSS
10_{108}	DDDDEEUNUWWDDDWWUUESSSUUNNNNWDDEEESSSUWWWWNNEENNDDSSSWUUUUEE
10_{109}	DDDDEEUUUNNNWWSUUESSDDNWWWDDESSEENNNUUUNWWSDDSSWDDENEUUEEEUUWWWS
10_{110}	DDDDEEUUUSSSWWNDDENNNWWSSWUUEEENUUSSDDDSWWNNNWDDESEUUEEEUUWWWN
10_{111}	DDDDEEUNNNUUSSDWWWSSEEEUUSSDDNDNWNNDEEUUUSSSWWNNNNDEEESSWWUUWW
10_{112}	DDDDEEDNDWWUUEEEUUWWWWSSEEEDDDWNNNUUUUSSSSDDNNNWNNEESSSUUUWW
10_{113}	DDDDEEEENNUWWWSSSSUUNNNEEEDDWDNNUUSSSSWWWWNNEEEDDDSSUUNUUUWW
10_{114}	DDDDDEEUUUUWSWWNNENNDDSWSSSEEEENNWWWDWWUUEEESSSDDNNNNUUWWUUSES
10_{115}	DDDDEEENNUUWSWWWSSEEEENNDWDDSSUUSSWWNNENNNUUSSSSDDDEENUNWUUUWW
10_{116}	DDDDEDNNUUUUSSSWWWNDDDENEEEEUUWWWWSSWWNNEDESSEENNWNNEESSSUUUWW
10_{117}	DDDDDEEUUUSSSWWNNENNWWWDDEEUNUUSWSSEEEEDDWWWWNNNDEESSSSUUUUNWN
10_{118}	DDDDDEEUUWNNNUUSSSEESDDWWWWNNNUEEEEESSWSUWWDWNNEESSSDDNNNEUUUUWWS
10_{119}	DDDDEDDWWUUUNNUUSSSSDDNEENNWNDDSSSUSUUNWWWNNEEEENDDSWWWSUUUUEE
10_{120}	DDDDEEDSSUUNUWWWNNEESEEEDDWSWWDNNUEUUUWSSUEEDDDNNNWWSSSSUEUNNUWW
10_{121}	DDDDEDNNUUUUSSSSDDNNNENNWWSSWWUUEEDEEESSWWWWNNNDDESEESUUUUWW
10_{122}	DDDDDEEUUSUUWWWWDDEEESSDDNNUNNNUUSSSSSWWNNNNDEEEDESSWSWWUUUUNN
10_{123}	DDDDEEEENNWWWUUUSSSSDDNNNDDDEEUUUUWWWWSSEEEDDDNNNNUUSSSUUUWW
10_{124}	DDDDEDSSUUUNNNNWWSSDDSEEEUUUWNNWDDWWSSEEENEUUUWW
10_{125}	DDDEEEUNNWWSSSUEEDDWWDDNNUUWUUESEDSSDDNNWWNNUEESUUUSWW
10_{126}	DDDEEEUUSWWNNWDDWNNEESSDDEEUUUWWSSWWNNNEDDEESSSUUNUUWW
10_{127}	DDDDEEEUUNWWSSWUWNUNNDDSSSSEUENNNWWWDDEEESSEEENNWWSUUUWW
10_{128}	DDDDEEEUUWWUNUEEDDWWWDDDSSSUUENNNNWDWSSSEEEUNNNUUSSUWW
10_{129}	DDDDDEESUUUUNWNWWSSDEDENEESSWWDNNNNUUSSSSWDWNNNEEEUUUWSW
10_{130}	DDDDEEEUUWWDNNUUSSSSWWNNNEDEEDSSSUUNNUUWWSDDDEEEUUWWNW
10_{131}	DDDDDEEUUWNNEESSSWWWWNUNEUEEDDDWSSSUUUNWWNNNEDESSSWSUUNN
10_{132}	DDDDEEESUUUWWWWNNEEESDSDDDNNUUUWSSSEEENNDWWWSWUSUUNN

K	**NEWSUD Form**
10_{133}	DDDDEEEUUUUSSDDNWWWWNNEDESSSUEUNEENNWWWSSSSEDDNNNUUUWW
10_{134}	DDDDENEUUUWWWSSSEENNDNWDWWSSUUUEEDDSWWWNNEEDSEENNNUUUSSW
10_{135}	DDDDEEUUUSWWWNNEENNEESSSDDNNWUUUWWDDESSSEEENNWWDWSSWSUUUNN
10_{136}	DDDDEDSESUUUWNNNWWWDDEEEUSSSWWNNNNUUSSSEEEDDDWWUUSUUNN
10_{137}	DDDDDEENEUUWWWWWUSSEEENNENDDSSSUWWWNUNEUESSSDDDNNUEUUUWW
10_{138}	DDDDEEUUNUWWWWSSEEEUNNNDDSSUEUUWWWDDDSEEEEDNNWWSSSUWUUNN
10_{139}	DDDDEENUUUWWDDWWWUUEESSEDENNNWWSSSDEENNNWWWUSSUUEE
10_{140}	DDDDEEUNUUNNNDWWDSSESSSWWNNUUEENNDDDSSWUUESEENNWWWUUSS
10_{141}	DDDDDEEUUNNWUUSSSWWWDDEEDENNEEUUWWWWSSSDDNNNEUESUEUUWW
10_{142}	DDDDEEEUUWWNWWNDDSSSEEEUNNUUWWWSSEDEDNNDDWWWUUEENUUUSS
10_{143}	DDDDEEEEUUNWWWSSWWDNNEESSSEENNNNUUSWSSDDDDNNUUSUEUUWWW
10_{144}	DDDDEEENNWUWSSEEUSUWWWWNNEEESDSDWDSSUUNNNNEDDDSSSUUWWUUUNN
10_{145}	DDDDEEEUUNWWWDDWWUUUEEENEDDSSSWWWNNNEESSUSWWNNNUEUSS
10_{146}	DDDDEDNNUUSSSSUUNNNEDDDWWWWSSUEEDEEUUWWWSDDNNUUNEESEUUWW
10_{147}	DDDDEEEUUNWWWWSSEEEEDDWWUSUUNNNENDDSSSSDDWNNNUUSUEUUWW
10_{148}	DDDDEENUUUNNWWDWSSWDDEEUUESEENNWWUSWWWNDDESSSEENNNWUUUSS
10_{149}	DDDDDEEEUUNUWWSDSSUUNNNNDDDSSSDEDNNUUSSUUWWWNNDEEESUUUWW
10_{150}	DDDDEEUUUWNWWSDSSDDNNUNNDEESUSSWWWUUEEEDNNWWWDDEEUNUUUSS
10_{151}	DDDDDEEUESSWWNNNNUUUSSSSDDNNUNEEDDDSSWUUUWWWDNNEEESUUUWW
10_{152}	DDDEEENUUWWSSSWWDDNEEEUUUWWDDSDDNNENUUSSSEDDNNNUUSUUWW
10_{153}	DDDDESEUUUWWDDDWWUNNEEESSSWWNNDNNUUUSSDSSEENNNWWWUUEES
10_{154}	DDDDEEEUNNWUUUSSDDDNDEEUUUWWWWNNEEESDSWWWWSSEENDNNUUUWSS
10_{155}	DDDDEEUUSSSWWNNENUNNDDSSSEEUUWWWWDNNEEEDEDSSSUWWUUUNWN
10_{156}	DDDDDEENUUUWWWWSDDENEESSSUUUNNNNDDSSSWWWWNNEESUSEEEUUNWW
10_{157}	DDDDDEEUUUSWWWWDDEEENNNUUUSWWSSSDDNNNEEEESSWWNUNWNUUSS
10_{158}	DDDDEEUUUNWWWWDDEEESSSWWUNNNNUUSSDSEENDNEESSWWWDWWUUUEEN
10_{159}	DDDDDEEUUUNWWWWDDEEUESUSSDDNNNNUUUSSSWWNNNDDSSSEEEUUUWWN
10_{160}	DDDDEEEUUUSWSUUNNNDDWSSSDDNNNUEEESSWWWWUUEEESDDDWNNUUUWW
10_{161}	DDDEEEDDWWWUSUEEEUNNUWWSSSDDDNNNUUSSESDDNNNUUSUUWW
10_{162}	DDDDEEEEENUUUWWWSSEDDDDNNUUUWSSSEENNNNEEDSSWWNNWWSSEUUUWW
10_{163}	DDDDDEEUUUUWNWDWSSDEEEENNWWWWSWUUESEESDDDNNNNUUSSSWSUUNN
10_{164}	DDDDDEESUUUWWWNNEEEEESDDWWSSWUUEUNNNNDDSSSSEENENNWWSUUUWW
10_{165}	DDDDDEEEUUUNWWWWSSEESDDNNNUEEESSWWWWSUUNENNNDDDESSUUUUWW
Granny	DDDDEEUESSWWNNNDEDSSUUUUWNNWWSSDEENEUUWW
Square	DDDDEDNNUUSSSWWDNNEEEUUUWWDWSSEENNNWUUSS