The DNA of Snakes

Md. Shahnawaz Khan[1] **and Walter D. Potter**[2]
[1,2]Institute of Artificial Intelligence, University of Georgia, Athens, Ga, United States

Abstract - *The Snake in the Box problem is an NP-Hard problem. The goal is to find the longest maximal snakes (a certain kind of path satisfying particular constraints described as "spread") in an n-dimensional hypercube [8]. With increasing dimensions the search space grows exponentially and the search for snakes becomes more and more difficult. This article identifies an underlying pattern among the known longest snakes in previously searched dimensions, which resembles the DNA of living cells in many ways. Surprisingly, these generic structures are fundamentally different for the four combinations of odd and even dimension and spread. It briefly explains the reason why they have different underlying structures. In odd dimensions with odd spread, there is one symmetric point and a unique mapping of complementary transition pairs and are discussed in detail in this paper. This article focusses only on one of these – odd dimension with odd spread. Later, it also reports three new lower bounds that are established using these generic structures from previously known longest maximal snakes. Another known longest snake in another odd dimension with odd spread is also found using this approach.*

Keywords: Snake-in-the-box, Generalization, DNA of snake, new lower bound, higher spread, longest maximal snake

1 Introduction

A snake is a special type of path in a graph (an n-dimensional hypercube) which does not violate its distance constraint described using the concept of "spread". Spread, being a concept of distance, is a non-negative number, and generally starts for spread k equal to 2. For spread 0, it has no meaning as technically it makes no contribution to the constraint. For spread 1, it simply requires a non-overlapping path traversing the n-dimensional hypercube and could be seen very similar to a Traveling Salesman Problem (often used as a standard problem in current AI literature). For spread 2 onwards it starts getting trickier and more computationally intensive to find such paths. The snake refers to the specific sequence of nodes in a graph and the edges joining these nodes form the path. While traversing it maintains the constraint that if the distance between any two nodes along the path is less than or equal to the spread then the shortest distance (Hamming distance) between them is equal to this distance along the path. For example, if node 0 and node x are placed like 0, _, x (where "_" could be any other node and node "x" is constrained) in a spread 2 or higher spread snake, then node x has to be a node which is exactly 2 Hamming distance away from node 0 (i.e. node x differs from node 0 in exactly 2 bits). If the distance between the two nodes along the path is greater than the spread then the shortest distance between these two nodes is greater than or equal to the spread. For example, if node 0 and node x are placed like 0, _, _, _, x in a spread k snake (for spread $k \leq 3$) then node x has to be a node which is at least k Hamming distance from node 0 (i.e. node x differs with node 0 in at least k bits). The maximally longest snake refers to the longest snake that can be found in a particular dimension-spread and cannot be grown further. So a path through nodes 0, 1, 3, 7, 6 would be the longest maximal snake of length 4 (distance between first node and last node in the path) in dimension 3 with spread 2.

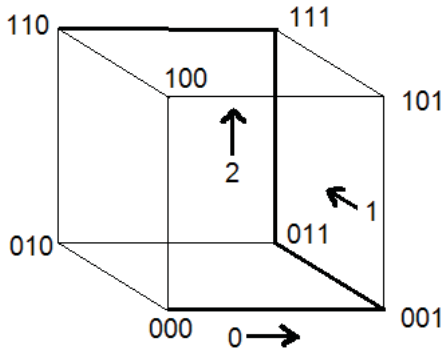

Figure 1: A spread 2 snake in a 3-dimensional hypercube - Snake (3, 2)

Snakes have been represented in various forms. Node-sequence representation, being the most naive and primary form of representation, is nothing but the ordered sequence of nodes that are traversed in an n-dimensional hypercube along the path (previously mentioned 0, 1, 3, 7, 6 is one such node-sequence representation). Among various other representations of snakes, transition sequence is a simple and parsimonious representation. For a 0-based transition sequence representation, it is a non-negative integer describing the transition of nodes (the position of change of the bit between the previous node and current node when the nodes are represented in a binary code) to build a snake. The change of node 0 to node 1 can be represented by transition "0" (node 000 changes to node 001 by changing the bit at position 0). Likewise, the traversal of node 1 to node 3 can be represented

by transition 1 (node 001 changes to node 011 by changing the bit at position 1). In short, the node sequence 0, 1, 3, 7, 6 can be written as 0, 1, 2, 0 in transition sequence. For any transition sequence, only the first node needs to be chosen but due to the symmetric nature of a hypercube any node would serve the purpose by naming it as node 0. A canonical snake, in a transition sequence representation, is a snake transition sequence such that the first occurrence of any transition precedes the first occurrence of any other transition that is bigger than it. For example, a snake starting as 0, 1, 2, 3, 1, 0, 4 would be a canonical snake, since the first occurrence of transition "0" precedes the first occurrence of all other transitions that are bigger than it and so on and so forth for all the other transitions in it. While a snake starting as 0, 1, 2, 4, 0, 3 would not be a canonical snake since the first occurrence of transition "3" does not precede the first occurrence of transition "4". This transition sequence (transition sequence 0, 1, 2, 4, 0, 3) can be represented in its canonical form by using the smallest unused transition for the first occurrence of every new transition while rewriting it (and using this replacement elsewhere). So for 0, 1, 2, it would still be 0, 1, 2 in its canonical form. When we encounter transition "4" we use the next smallest unused transition "3" for it (and replace "4" with "3" everywhere else). So the sequence 0, 1, 2, 4, 0 would become 0, 1, 2, 3, 0. Later when we encounter a new transition "3" we have to use the next unused smallest transition i.e. transition "4" (and replace transition "3" in the old sequence with transition "4" in its canonical form). So its canonical form would be 0, 1, 2, 3, 0, 4. Also previous works have shown that a canonical representation of transition sequence can be used to represent any snake [4].

The snake in the box (SIB) problem has been an interesting and challenging problem for both mathematicians and computer scientists [8]. The challenge has been taken to another level every time a particular dimension's longest maximal snake(s) are found, as the search space grows exponentially. As the search space grows exponentially, it gets more and more difficult to do an exhaustive search and some kind of heuristic is required. David Kinny mentions some complete search techniques and illustrates the role of branching factor while backtracking [1]. He mentions the crucial pruning of the search space by using a canonical form [2].

2 DNA Basics

In this section, some basic and generic information about DNA is discussed which will help the reader to follow and appreciate the similarities discussed in the latter sections. Deoxyribonucleic acid or DNA is a double-stranded helix, with the two strands connected by hydrogen bonds [2] [3]. Its structure is shown in Figure 2.

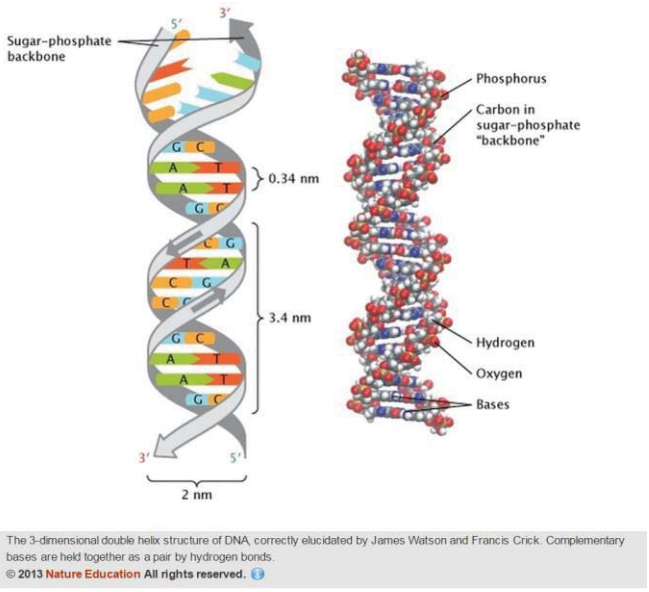

Figure 2: A double helix structure of DNA* [9]

Courtesy:
http://www.nature.com/scitable/topicpage/discovery-of-dna-structure-and-function-watson-397

*The referenced web page was visited on November 2, 2014

It is found in every living cell and encodes the genetic instructions used in various aspects of development and functioning of living organisms. DNA controls the growth, functioning and reproduction of cells in the living organisms. The information in DNA is stored as a code which is made up of four chemical bases: adenine (A), cytosine (C), guanine (G), and thymine (T). The order, or sequence, of these bases determines the information available for building and maintaining an organism. These DNA bases pair up with each other, A with T and C with G, to form units called base pairs. The base pairs are constant, i.e. base A would always pair up with base T and base C with base G. It is beyond the scope of this article to discuss the reason why these bases always pair up with each other.

3 Building Canonical Snakes

The canonical snakes are representative of all the snakes in the search space or in other words all the snakes in the search space can be represented using one of the canonical snakes. We first introduced an exhaustive search algorithm to build canonical snakes in transition sequence as shown in Table 1. This algorithm is the first known algorithm to validate a snake in transition sequence representation without converting it into any other form. The validating algorithm is based on the idea of number of unpaired transitions that helps

in maintaining the snake spread-k constraints. The exhaustive search makes no assumption about its search space. It tries a transition by adding it to a snake and validating the sequence. If it succeeds it moves to search for the next transition else it tries another transition until all the transitions available have been tried, after which it backtracks to its last successful transition and tries another transition from it. This is repeated until all the transitions at the first position have been tried and there is no other backtracking possibility.

Table 1: Building Snakes for Dimension n Spread k

- Initialize an ordered list (for transition sequence), call it the Primary List (**PL**) and 2 auxiliary sets (Paired and Unpaired Transition Set – **PTS** and **UTS**, which are mutually exclusive)
- Initialize **PTS** with transitions 0 to n-1 // 0 based transition sequence
- Add transitions 0 to k in the **PL** and in the **UTS** and remove these transitions from **PTS**
- Set a flag **isValid** to **true**

  While (number of elements in **UTS** >= **k** and flag **isValid**)
  {
  - Add an element i to **PL** which is different from last **k** transitions* and is a member of {0,..,**n**-1}
  - Flip the membership of this element i between **PTS** and **UTS** // mutually exclusive
  - If (number of elements in **UTS** >= **k)**
    {
    - Copy the **PTS** and **UTS** to new temporary sets Temporary PTS (**T-PTS**) and Temporary **UTS** (**T-UTS**)
    - Starting from the first transition in PL, flip the membership of each transitions between the current **T-PTS** and **T-UTS** and in each step check the number of elements in **T-UTS**>=**k** else set the flag **isValid** to **false** and break from this loop

    }
  - Else
    {
    - Set the flag **isValid** to **false**

    }

  }
- If (not **isValid**)
  - Remove the last added transition

*Pruning the search space by removing certain invalid snakes

In an n-dimensional hypercube, for a 0-based transition sequence, the transition sequence consists of numbers between 0 to n-1. The unpaired number of transitions maintains the spread in the path. So, if we are looking at a sequence 0, 1, 2, 3, 1, we see that there are two transition "1" (in other words paired), while 0, 2, 3 are unpaired. As the number of unpaired transitions drops below k, the k-spread constraint is violated. For any transition chunk of length greater than or equal to k, it should hold that there are at least k unpaired transitions. And for any transition chunk of length d less than k there should be at least d unpaired transitions. So, {0, 1, 2, 3, 1}, {1, 2, 3, 1}, {2, 3, 1} and {0, 1, 2} are some of the examples of such transition chunks.

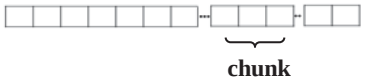

Figure 3: A transition chunk

As shown in Table 1 for a spread k snake, choosing a transition different from the last k transitions prunes the search space by removing the invalid snakes. An extra pruning step that is added is that if the next element that is being added to the list matches with the element which is at the last $(k+1)^{th}$ position in the list then the transition at the last $(k+2)^{th}$ position should not be there in the last k-transitions. For example, consider a transition sequence as $\{a_{k+2}, a_{k+1}, a_k, a_{k-1}, \ldots, a_1\}$ consisting of k+2 transitions. If we want to add transition "a_{k+1}" as the next transition in this transition sequence then we can add it only if transition "a_{k+2}" is not there in the subsequence $\{a_k, a_{k-1}, \ldots, a_1\}$.

4 The DNA

Snake 1 (11, 5): [0, 1, 2, 3, 4, 5, 6, 7, 1, 2, 8, 5, 9, 7, 0, 10, 8, 1, 4, 5, 7, 6, 10, 8, 3, 4, 2, 5, 10, 9, 6, 4, 1, 5, 7, 0, 9, 6, 3]

Snake 2 (6, 2): [0, 1, 2, 3, 1, 0, 4, 3, 0, 5, 4, 0, 1, 3, 4, 0, 2, 4, 1, 0, 4, 3, 1, 5, 3, 4]

Snake 3 (7, 3): [0, 1, 2, 3, 0, 4, 5, 1, 0, 3, 6, 4, 0, 1, 2, 3, 0, 4, 5, 1, 0]

4.1 The Underlying Structure

The three snakes shown above have a few things in common. Apart from being the longest maximal snake in a particular dimension-spread, they also share a particular underlying structure upon which it is built. Snake 1 is a spread 5 snake in dimension 11. Snake 2 is a spread 2 snake while Snake 3 is a spread 3 snake. All these snakes are the longest maximal snakes and are canonical palindromes (a canonical snake whose reverse when expressed in a canonical form is equal to the original canonical snake). They have one or two points of symmetry based on if they have an even spread or an odd spread. Let us look at the same snakes again with the highlighting.

Snake 1 (11, 5): [0, 1, 2, 3, 4, 5, 6, 7, 1, 2, 8, 5, 9, 7, 0, **10, 8, 1, 4, 5, 7, 6**, **10, 8,** 3, 4, 2, 5, 10, 9, 6, 4, 1, 5, 7, 0, 9, 6, 3]

Snake 2 (6, 2): [0, 1, 2, 3, 1, 0, 4, 3, 0, 5, 4, 0, 1, 3, 4, 0, 2, 4, 1, 0, 4, 3, 1, 5, 3, 4]

Snake 3 (7, 3): [0, 1, 2, 3, 0, 4, 5, 1, 0, 3, 6, 4, 0, 1, 2, 3, 0, 4, 5, 1, 0]

The shaded region highlights the basic structure of the snake which lays the foundation of a particular snake, similar to specific sequencing of genes in DNA which later decides everything for the organism. This shaded region, which is termed as the DNA, is defined as "*DNA of a valid snake is the smallest portion of the snake (approximately at the center of the snake) that contains all the possible transition sequences for the snake and has one or more points of symmetry. It also defines the complementary pairs of the transitions that should be used in the remaining parts of the snake."* There is one or more than one point of symmetry in the DNA. In the simplest case, where there is only one point of symmetry, the equidistant transitions to the left and right of this symmetric point occur in pairs and are called complementary pairs henceforth in the paper. These complementary pairs always occur in pairs to the left and right of the DNA throughout the snake.

4.2 Odd and Even Dimensions

Odd and even dimensions have different DNA in their longest snakes, primarily because the number of possible transitions in the two types of dimensions is different, i.e. for odd dimensions it is odd, while for the other it is even. In an odd dimension, the arrangement of a possible odd number of transitions for pairing in the underlying structure (similar to the base pairing in DNA) will be different than the even transitions where the number of possible transitions is even. The spread of the snake also plays a role in defining the structure of the DNA. The number of initial transitions that are used in the DNA (shown in red color) is equal to its spread (since no k transitions can be the same in a spread k snake). These initial transitions form the core of the DNA. For odd spread, an odd number of transitions is already used in the DNA to form its core. Now, for odd spread snakes in an odd dimension, the remaining transitions that have to be paired uniquely, after forming the core, are even in number and can be uniquely paired. But for such snakes (snakes with odd spread) in even dimensions, the remaining transitions are odd in number and cannot be uniquely paired. Let us take an example of snake (7, 2) to illustrate more on the pairing of complementary pairs and symmetric points in DNA.

Snake (7, 2) : 0, 1, 2, 0, 3, 1, 0, 4, 2, 1, 0, 3, 5, 0, 1, 2, 4, 0, 6, 5, 0, 4, 2, 0, 3, 4, 0, 1, 2, 4, 0, 3, 5, 0, 4, 2, 0, 3, 4, 0, 1, 2, 0, 6, 1, 0, 4, 2, 1, 0

The snake shown above is the longest maximal snake in dimension-spread (7, 2) [7]. Since for spread 2 snakes no two consecutive transitions can be the same, transitions 3 and 4 appear in the middle as shown using red color. The remaining 5 transitions have been paired but not uniquely, most of the transitions have been paired with more than 1 transition in the DNA (shown as the highlighted grey area). Also since there is more than one point of symmetry their pairing varies for three ways of finding the point of symmetry, i.e. {(3, 4), (3), (4)}. Say for example "5" can be paired with "0" if "3, 4" is the point of symmetry as both are equidistant from this point of symmetry. "5" can be paired with transition "3" if "4" is the point of symmetry. "5" can also be paired with "4" if "3" is the point of symmetry. The DNA for even spreads is difficult to create and we will restrict ourselves to the odd spreads. As explained earlier, for odd spreads in even dimensions the remaining transition options for creating the DNA would be odd which again would create non-unique pairing. To simplify our task we will confine ourselves to odd dimensions with odd spread. The remaining dimension and spread combinations are intended to be pursued as future work.

4.3 Similarity with DNA

So how is the underlying structure similar to DNA? And what role do these subsequences play in building snakes? If we observe closely we will find that all the transitions {0… n-1} have been used in creating this shaded part. Similar to the DNA in living cells, it contains all the information/ingredients that could be used later. Apart from having all the transitions it also defines two more interesting features, the base-pairing and the length of the longest snake possible that can be grown using this underlying structure. The first feature is easier to explain and demonstrate while the second feature can only be explained from the results obtained as is the case with mapping of particular genes to a particular characteristic in a living organism (i.e. mapping genotype with phenotype). Similar to the base pairing in DNA, i.e. base A always occurs with base T and base C always occurs with base G, the transition sequences also always occur in pairs defined using this underlying structure. In other words this underlying structure decides the transition that would appear with its complementary transition at any two equal distances from the symmetric point. Let us take the example of Snake 1, we see that the distance of transition "7" on the left side of transition "5" (the symmetric point), is always the same as the distance of transition "4" on the right side of transition "5" and vice versa. This is what also makes it a canonical palindrome (a canonical snake whose reverse when expressed in a canonical form is equal to the original canonical snake).

4.4 Building the DNA

Let us start from scratch while rebuilding these underlying structures for snakes. Building upon the idea from the previous section (Section 2.3), which described basic rules for a canonical snake in a transition sequence representation, we have the following mandatory guidelines:

1. No k subsequent transitions can be the same in a spread k snake.
2. In the snake, for all subsequences of size greater than the spread, the number of unpaired transitions is greater than or equal to the spread.

Let us build the DNA of Snake 1, the DNA of the longest snake in dimension 11 with spread 5. Let us start from the symmetric point (for odd spreads there is one symmetric point). So for keeping it simple, let us use "0" as the symmetric point. Now since no adjacent k (k is 5 in this case) sequences can be the same we can put four other transitions in this structure as shown below.

3, 1, 0, 2, 4

The order of transitions does not matter as this is the defining stage where the pairs are being defined and whatever transition we decide to put would form the definition of pairing. We could have used {3, 1, 0, 2, 4} or {0, 1, 2, 3, 4}. We built the first sequence by adding "1" to the left of transition "0" and "2" to the right of "0". Then we added "3" to the left and "4" to its right. From the above sequences we have defined that transition "1" is paired with transition "2" and transition "3" is paired with transition "4" as they are at equal distance from "0" on the left and right side. So far, for spread k, if the dimension n is equal to k then putting all the transitions like this would make the longest snake of length k. But as we increase the dimension, we need to decide where the other extra transition sequences would be placed. In our example the next two transitions (say transition "5" and transition "6") we can have the following sequences where either each of these transitions is placed in the same way to each side of the structure or switched on the other side as shown:

6, 5, 3, 1, 0, 2, 4, 5, 6 or 6, 5, 3, 1, 0, 2, 4, 6, 5

Both of these would be the longest snakes for dimension-spread (7, 5). As we go higher in the dimensions, we start adding new transitions or reusing the previous used transitions to left and right (if these transitions do not make the snake invalid). Based on the structure of the longest snake found so far, the second one containing {6, 5,…., 6, 5} is more common in odd dimensions with odd spread. So, let us add the next two transitions to this sequence, as shown:

7, 6, 5, 3, 1, 0, 2, 4, 6, 5, 8

After adding these, we can re-use the pair of transitions 2 and 1. One of the common patterns that have been found is that during reusing the transitions the transition that was placed on the left side last time is preferred on the right side and vice versa. So the new structure would look like:

2, 7, 6, 5, 3, 1, 0, 2, 4, 6, 5, 8, 1

At this point adding the remaining transitions (transitions 9 and 10) would look like:

9, 2, 7, 6, 5, 3, 1, 0, 2, 4, 6, 5, 8, 1, 10

This is all we need for the longest snake. This is the DNA of the longest snake in dimension-spread (11, 5). This is the same structure as that of Snake 1. In fact, when we used this underlying structure to build the longest snake, we found the following snake whose canonical form is Snake 1.

Found: [7, 3, 10, 8, 1, 0, 4, 2, 3, 10, 5, 0, 9, 2, 7, 6, 5, 3, 1, 0, 2, 4, 6, 5, 8, 1, 10, 0, 6, 9, 4, 1, 3, 0, 2, 7, 9, 4, 8]

Canonical: [0, 1, 2, 3, 4, 5, 6, 7, 1, 2, 8, 5, 9, 7, 0, 10, 8, 1, 4, 5, 7, 6, 10, 8, 3, 4, 2, 5, 10, 9, 6, 4, 1, 5, 7, 0, 9, 6, 3]

The above snake is the longest known snake in (11, 5) and is of length 39. This is also the maximally longest snake in this dimension-spread and is confirmed through exhaustive search in dimension-spread (11, 5).

4.5 Working -Skin Nodes and Shadows

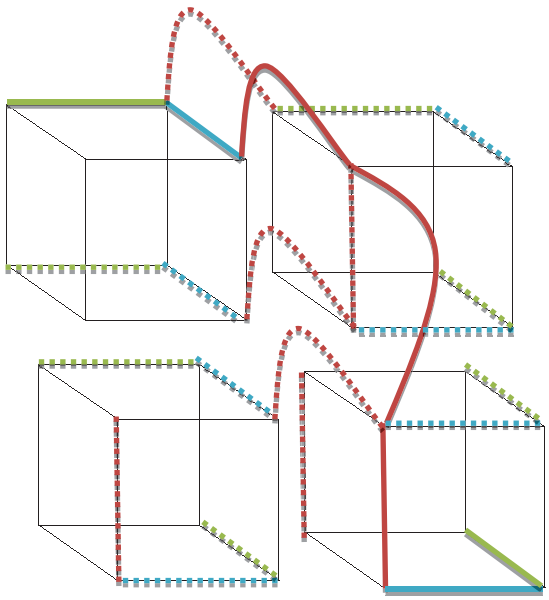

Figure 4: The longest snake and its shadows in dimension-spread (5, 3).

No hypothesis is ever complete without an attempt to explain its workings. In this section we will attempt to explain why this works. When a node is used in the path of a spread k snake, it makes all of its neighbors, at a Hamming distance of k or less, unusable for future path options (except the k-nodes in the path). These unusable nodes are called skin nodes. These skin nodes when joined in the sequence of being created by the snakes are termed as "shadows" in a smaller hypercube. Figure 4 shows a dimension 5 spread 3 longest snake and the

shadows (connection of skin nodes) it casts upon its composite smaller hypercube (i.e. Hamming distance = 1). In an n-dimensional hypercube, for spread k, when we traverse an edge of the hypercube, this edge casts its shadow in all the adjoining smaller hypercubes and is resonated until spread k. These shadows inhibit the growth of snakes in the future. But if an algorithm can strategically place the edges considering our future moves such that our paths are less and less affected by these shadows then a long snake would be possible. The pairing of transitions helps us in maintaining and strategically placing the shadows. Of course, choosing the correct pair is decided by both of the factors, the past transitions that have been used and the future transitions that are left unused. The complementary shadows pave the way for a snake to move inside these tightly packed shadows.

5 Results and Discussion

Once the DNA is chosen, we start assembling the pairs to the left and right side of the structure (DNA) while following the pairing rule (using complementary pairs on the left and right side of the DNA). After adding the complementary pairs on both sides the snake is validated. For validating the snakes faster, we store a map of unpaired transitions from each position for the current snake and update it when new pairs are added on both sides (following the same fundamentals described in Section 2.3). Picking up the complementary pair is done as an exhaustive search and we can call it an exhaustive complementary pair search. Though the DNA occupies a very small part of the snake and intuitively it seems that the search for the snake is still the same old difficult job, the reality is quite the contrary. First of all, by only allowing a unique pair of transitions from the DNA (rather than an arbitrarily large combination of transition pairs) we restrict the search space to a much smaller area. The second and the most important contribution of the DNA is that the right DNA lays the required foundational structure that can only grow to be the longest snake in the hypercube. One of the current limitations of this approach is that these structures are very simple only for odd dimensions with odd spread. For the other three combinations of odd and even dimension-spread, these structures are far more complex and become less analogous to the helical structure of DNA with unique base-pairing. We intend to pursue the research in the remaining types of dimension-spread combinations, but for now we confine ourselves to odd dimensions with odd spread.

Using this DNA structure we were able to find new lower bounds for the snakes in (13, 5), (15, 7) and (17, 7). The previous best known results are from [5] [6].The longest snake known so far for (9, 3), of length 63, was also found using this approach. The results are summarized in Table 2. The value in the parentheses in the right hand column is the previously known lower bound. The search in dimension-spread (15, 7) was completed by exhausting the complete search space defined by the structure which means no other longer snake is possible for the structure. For the other dimension-spreads the search could not be completed at present and longer snakes are possible for such structures.

Table 2: New Lower Bounds for Snakes

Dimension-spread	Lower bound
(13, 5)	85 (79)
(15, 7)	57[c] (55)
(17, 7)	103 (98)

c - Complete search for the given structure

6 References

[1] David Kinny. (2012). "A New Approach to the Snake-In-The-Box Problem," *Proc. 20th European Conference Artificial Intelligence*, 462–467

[2] F. H. Crick, J. D. Watson. (1954). "The Complementary Structure of Deoxyribonucleic Acid", *Proceedings of the Royal Society (London) A223*, 80

[3] J. D. Watson, F. H. C. Crick. (1953). "Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid", *Nature 171*, 737

[4] Kochut, K. J. (1996). "Snake-In-The-Box Codes for Dimension 7". *Journal of Combinatorial Mathematics and Combinatorial Computations* 20:175-185

[5] S. Hood, D. Recoskie, J. Sawada, D. Wong. (2011). Snakes, coils, and single-track circuit codes with spread k, *Journal of Combinatorial Optimization*, 1–21

[6] S. Hood, J. Sawada, C.H. Wong, (2010). "Generalized Snakes and Coils in the Box"

[7] W. D. Potter, R. W. Robinson, J. A. Miller, K. J. Kochut, D. Z. Redys. (1994). "Using the genetic algorithm to find snake-in-the-box codes", *Proceedings of the 7th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, 421-426

[8] W.H. Kautz. (1958). "Unit-distance error-checking codes", *IRE Trans. Electronic Computers*, 179–180

[9] http://www.nature.com/scitable/topicpage/discovery-of-dna-structure-and-function-watson-397