BUILDING SNAKES FROM DNA – A STEP TOWARDS GENERALIZING THE SNAKE IN THE BOX PROBLEM

by

MD. SHAHNAWAZ KHAN

(Under the Direction of Walter D. Potter)

ABSTRACT

The snake in the box problem is an NP-hard problem which has been a challenging problem for both computer scientists and mathematicians. It aims to maximize certain types of paths (snakes) in a graph, an n-dimensional hypercube while satisfying certain constraints described using the concept of spread. This thesis identifies a common pattern among the longest snakes which is very similar to the DNA in living cells both structurally and functionally. It introduces a radically new approach to use this underlying pattern (the DNA) for building the longest snakes in a generalized way. By using these structures in three different dimensions three new lower bounds are established beating the previously held records. This thesis also attempts to explain why such underlying structures contribute to the longest snakes and in general how the longest snakes arrange themselves in a hypercube.

INDEX WORDS: Snake-in-the-box, generalization, DNA, DNA of snake, new lower bound, higher spread, longest maximal snake

BUILDING SNAKES FROM DNA – A STEP TOWARDS GENERALIZING THE SNAKE IN THE BOX PROBLEM

by

MD. SHAHNAWAZ KHAN

B.TECH, AMU, INDIA, 2010

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2014

BUILDING SNAKES FROM DNA – A STEP TOWARDS GENERALIZING THE SNAKE IN THE BOX PROBLEM

by

MD. SHAHNAWAZ KHAN

DEDICATION

To my parents for everything.

ACKNOWLEDGEMENTS

I would like to thank Dr. Potter for his support and encouragement for all these years in the AI center. I would also like to thank Dr. Rasheed for his help and advises during my stay here. I also thank Dr. Maier for being in my committee and helping me whenever I had doubts. At the end, I would also like to thank people at the AI center for all the discussions inside and outside the classes.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION AND LITERATURE REVIEW

This thesis is about discovering an underlying general structure among previously established longest snakes and using this general structure to establish new lower bounds for snakes in other dimensions for the Snake in the Box problem. This radically new approach is primarily inspired from the role of DNA in living cells which contains the instructions for the development and functioning of the cell. The Snake in the Box (SIB problem) is a graph search problem which continues to baffle both computer scientists and mathematicians. The term “snake” describes a specific traversal sequence of nodes in a graph (a hypercube, whose size is described by its “dimension”) without violating certain distance constraints described using the concept of “spread”. SIB is an NP-Hard problem and the first works were published in 1958 by Kautz (Kautz 1958). The snake refers to the sequence of traversing through nodes in a graph such 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. 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. It finds its application in signal detection, error correction, modulation of signal codes etc. (Yehezkeally et al. 2012). Snakes in any dimension “n” with spread “k” can be represented using the notation “Snake (n, k)”. We will use this notation throughout all the remaining chapters. We will often refer to this dimension, spread combination as dimension-spread.

This thesis started as a motivation to find a common repeating pattern among the longest snakes known so far and be able to predict the longest snakes in other dimensions-spread by generalizing it. As the SIB problem is an NP-hard problem, the search space grows exponentially with each increasing dimension. For any brute force search technique the search space is so huge that the exhaustive search of larger dimensions, with current computer processing capabilities, will take years and decades and even might not be finished within a human lifetime. In recent years several heuristics have been applied as an alternative to the brute force exhaustive search (Potter et al. 1994). Generalization of an NP-hard problem would be too ambitious a project and machine learning algorithms came as an affordable solution. Among several approaches that were tried, a Reinforcement Learning approach was tried to learn the pattern and predict the transition sequences for building the longest snakes. Since the size of the available data for training is too small (i.e. the previously established longest snakes in several dimensions are of the order of 10^1) and was not sufficient enough for the model to let it generalize, therefore this project could not grow further. Later, while analyzing the data, the longest snakes collected for training the learning model, several common structures were found and were exploited to hunt for snakes in other dimensions.

These common structures form the basis of this thesis. The algorithms explained in the subsequent chapters which are for submission to conferences exploits this general schema. The importance of this generalization is emphasized by the results obtained, as three new lower bounds for snakes in several dimension-spread are established.

At the end (Chapter 5), the thesis report is concluded by highlighting the salient features of the algorithms discussed in the chapters. These algorithms define a particular way to build snakes and explain the more natural way these snakes arrange themselves in a hypercube. We also discuss some of the possible limitations of the algorithms and ways to overcome them. Being a fundamentally new approach, we also discuss the future applications of the algorithms to find newer lower bounds in dimension-spreads yet to be searched.

CHAPTER 2

THE DNA OF SNAKES [1]

[1] Md. Shahnawaz Khan and W. D. Potter. Submitted to International Conference on Artificial Intelligence'15, 01/30/2015.

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.

2.1 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 ≤ 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.

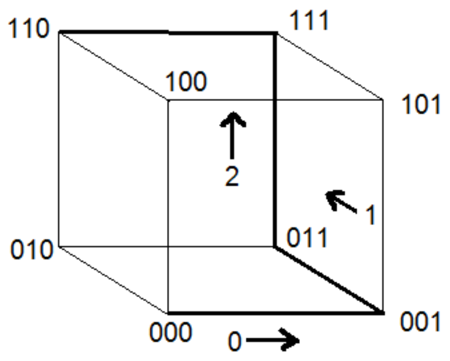


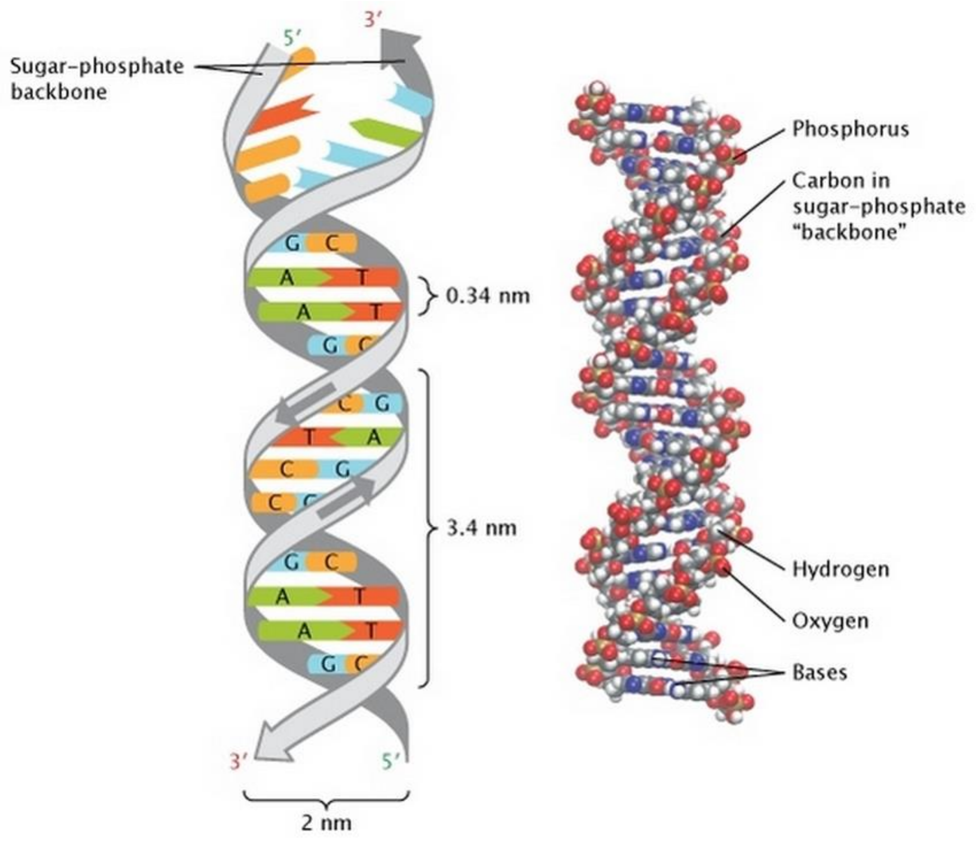
Figure 2.1: A spread 2 snake in a 3-dimensional hypercube - Snake (3, 2)

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].

2.2 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.



The 3-dimensional double helix structure of DNA, correctly elucidated by James Watson and Francis Crick. Complementary bases are held together as a pair by hydrogen bonds.
© 2013 Nature Education All rights reserved.

Figure 2.2: 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.

2.3 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 2.1: 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)
  {
  o Add an element i to PL which is different from last k transitions* and is a member of
    {0,..,n-1}
  o Flip the membership of this element i between PTS and UTS // mutually exclusive
  o 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
    }
  o Else
    {
    ▪ Set the flag isValid to false
    }
  }
• If (not isValid)
  o 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.

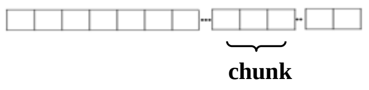


Figure 2.3: A transition chunk

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}, …, 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}, …, a_1}.

2.4 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]

2.4.1 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.

2.4.2 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.

2.4.3 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).

2.4.4 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, <u>3, 1, 0, 2, 4,</u> 5, 6 or 6, 5, <u>3, 1, 0, 2, 4,</u> 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).

2.4.5 Working -Skin Nodes and Shadows

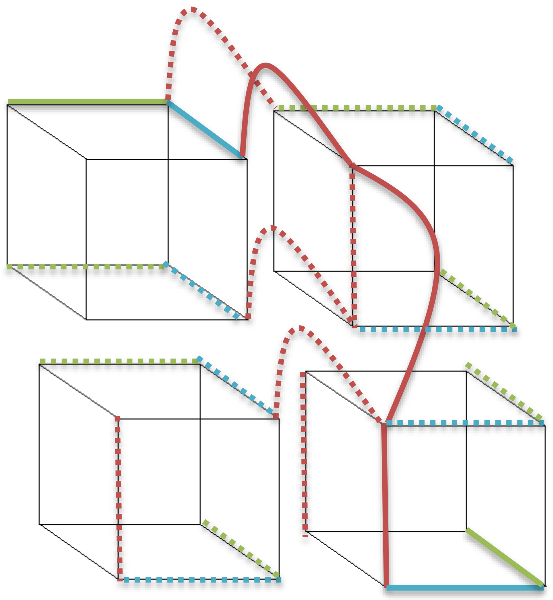


Figure 2.4: The longest snake and its shadows in dimension-spread (5, 3).

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.

2.5 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 2.2: 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

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

CHAPTER 3

COMPLEX DNA AND GOOD GENES FOR SNAKES [2]

[2] Md. Shahnawaz Khan and W. D. Potter. Submitted to International Conference on Artificial Intelligence'15, 02/17/2015.

ABSTRACT

The Snake in the Box problem deals with finding the longest snakes in an n-dimensional hypercube. The snake is supposed to follow a specific distance constraint, described by the term “spread”. It is an NP-Hard problem and searching the entire search space is not a feasible option as the search space grows exponentially with increasing dimensions. In the previous paper [1], a generic pattern among the longest snakes for the Snake in the Box problem was discussed. This generic pattern was termed the DNA because of the structural and functional similarity with the DNA of living cells. It is fundamentally different for each of the four combinations of odd and even dimension and spread. In the previous paper, we discussed the simplest pattern, found in the odd dimensions with odd spread. This paper illustrates one of the complex DNA patterns that is found in the other three types of odd and even combinations of dimension and spread. It also discusses several possible combinations of transition sequences in a simple DNA pattern (similar to gene combinations in the DNA of living cells) and their effect on the length of the longest snakes that can be grown from it.

3.1 INTRODUCTION

Snake in the box problem is an NP-Hard combinatorial problem which has been pursued by both computer scientists and mathematicians for several decades [8]. It aims to find the longest maximal snakes in an n-dimensional hypercube. A snake is a special type of path in a graph (an n-dimensional hypercube) that does not violate certain distance constraints, described using the concept of “spread”. A snake represents a path in an n-dimensional hypercube, such that if the distance between any two nodes along the path is less than or equal to the spread then the shortest distance (the Hamming distance) between them is equal to this distance along the path. If the distance between any two nodes along the path is greater than the spread then the shortest distance through the graph between these two nodes should be greater than or equal to the spread. Spread is nothing but a positive integer which represents this distance constraint and generally starts with 2. Several works have been published on the longest snakes for spread 2 and higher in several dimensions [2] [3] [7] [9] [10].

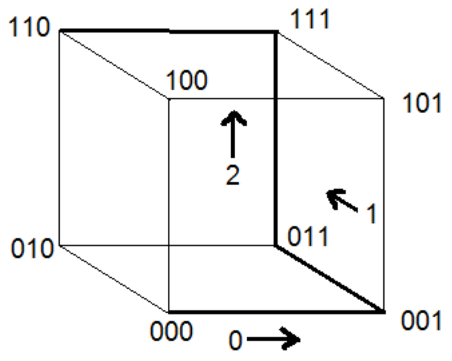


Figure 3.1: Transition sequences (0-based) in a 3-dimensional cube

The longest maximal snake for a particular dimension-spread refers to the longest snake that can be found in a dimension-spread and cannot be grown further. Snakes can be represented in several ways. Among various representations of snakes, the transition sequence is a simple and parsimonious representation. For a 0-based transition sequence representation, a non-negative integer describes 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 (see Figure 1). 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 0, 1, 2, 3 is a canonical snake in dimensions greater than or equal to 4 while 0, 1, 3, 2 is not a canonical snake as the first occurrence of transition "2" does not precede the first occurrence of transition "3" in the second case. Given the current computing resources, it is not possible to search the complete search space of the graph to find the longest maximal paths in dimensions greater than 7 [5]. Often several heuristics are applied to hunt for these snakes in the hypercube [7]. This paper discusses three things in a broad sense. First of all it briefly explains the fundamentals relating core subsequences to DNA [1]. It distinguishes the two types of DNA based on the mapping of complementary pairs and terms them as simple and complex. Later it explains the more complex DNA and how it is used in the snake. Finally, the last topic covered is about how to build simple DNA that would grow to long snakes.

3.2 DNA OVERVIEW

In [1], a generalized underlying structure was explained which was found to be common among the longest snakes in several dimension-spreads known so far. It also discussed the similarity they draw with the DNA of living cells. These underlying structures form the basic foundation in the potential construction of the longest snakes. Using these structures three new lower bounds of snakes in three separate dimension-spreads were found. This structure termed as the DNA was 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.*" One of the important characteristics of the DNA was that they have all the possible transitions that can participate in building the snake present in them. They also define the pairing of transitions that should follow in the remaining part of the snake. A concept of shadows was used to explain the underlying behavior while using the DNA structure in building snakes.

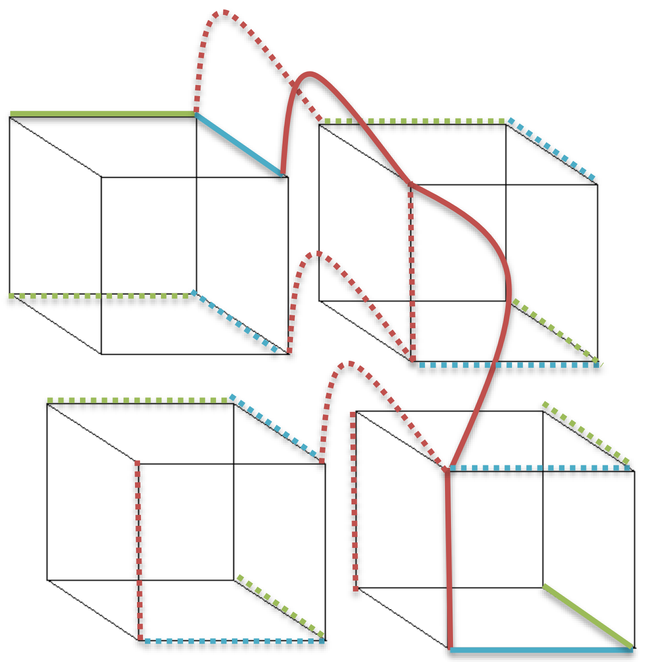


Figure 3.2: The longest snake and its shadows in dimension-spread (5, 3)

Also in [1], the broad differences in the DNA structure of odd and even dimensions were discussed. The discussion later was restricted to the odd spread as even spreads had multiple symmetric points, leading to multiple mappings of complementary pairs. For odd spreads, which have a single symmetric point, the even dimensions were also excluded since the remaining number of possible transitions in such dimensions, apart from the odd number of distinct transitions required to form the core of the DNA, were odd and could never have a unique mapping. In short, odd dimensions with odd spreads were the only point of discussion. Here we categorize the DNA into two, based on their unique or multiple mapping of complementary pairs. They are termed:

1. Simple DNA
2. Complex DNA

Simple DNA (DNA in odd dimensions with odd spreads) was the primary focus of the DNA discussion in [1]. In this paper we will try to explain our complex DNA. We will discuss the possible structures and will highlight some of the structures containing good genes (the transition pair combination that leads to longer snakes).

3.3 DECODING COMPLEX DNA

For snakes whose spread is even or for even dimensions with odd spread, there are many reasons for non-unique mapping of complementary transition pairs. One such reason could be that there is more than one symmetric point in the DNA for even spread snakes. Also for cases where odd numbers of transitions are left to be used for defining the complementary transition pairs (after forming the core of the DNA), these odd number of transitions lead to a non-unique mapping. For these types, the DNA is more complex than the one for odd dimensions with odd spread. In this section we will try to explain some of these complex DNAs with examples. For our first illustration, let us take the first example as the longest maximal snake in dimension 7 with spread 2, which is of length 50 and is shown below.

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

This is the longest maximal snake of dimension 7. The DNA of this snake is shown in the shaded grey region while at its core the two transitions (transitions “3” and “4”) are shown in red color (since spread = 2). Let us carefully examine the remaining part of the snake. We start with the complementary transition pair mapping, the mapping of the transition pairs equidistant on the left and right side of the symmetric point(s), which is defined in the DNA for this snake. For the snake shown above, when {3, 4} are together considered as the symmetric point, the complementary pairs of transition “0” are “5”, “4” and “0” and are shown below with superscripts.

$6, 5^{c1}, 0^{c2}, 4, 2, 0^{c3}, 3, 4, 0^{c3}, 1, 2, 4^{c2}, 0^{c1}, 3$

(“3” and”4” as the symmetric point)

The distance of complementary pair c1 is 5 from the symmetric points while the distance of complementary pair c2 is 4. The distance of complementary pair c3 is 1. The other two complementary pairs that appear in the snake are “0” with “6” and “0” with “3”. These complementary transition pairs are defined when transitions “3” and “4” are considered as the symmetric point individually, as shown.

$6^{c4}, 5, 0, 4, 2, 0, 3, 4, 0, 1, 2, 4, 0^{c4}, 3$

(“3” as the symmetric point)

$6, 5, 0, 4, 2, 0, 3^{c5}, 4, 0^{c5}, 1, 2, 4, 0, 3$

(“4” as the symmetric point)

In short, in the DNA we observe that “0” forms the complementary pairs with “5”, “4”, “0”, “0”, “4”, “6”, “3”, “4” and “0” on the left and right sides of the DNA. All the other complementary transition pairs that have been used in this snake are shown below with their names as superscripts. The transitions forming the pair share the same complementary pair name on the left and right side of the symmetric point (such as “c6”).

6, 5, 0, 4^{c6}, 2^{c7}, 0, 3^{c9}, 4^{c9}, 0, 1^{c7}, 2^{c6}, 4, 0, 3

(“3” and”4” as the symmetric point)

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

(“3” as the symmetric point)

6, 5^{c8}, 0, 4, 2^{c10}, 0, 3, 4, 0, 1, 2^{c10}, 4, 0, 3^{c8}

(“4” as the symmetric point)

The only complementary pair that cannot be defined using this DNA is that at three places “1” is paired with itself. One of the possible explanations that can be accommodated for this anomaly is that while adding these complementary pairs, since the symmetric point also changes based on the complementary pair we are choosing so the DNA includes one neighboring transition to its left or right to keep the DNA always symmetric about its symmetric point. So say if we are using complementary pair c’ where the symmetric point is “4” and the pairs are “0” and “1” as shown:

6, 5, 0, 4, 2, $0^{c'}$, 3, 4, 0, $1^{c'}$, 2, 4, 0, 3, 5

As “4” is the new symmetric point transition, so it should include “5” on its right in the DNA to make it symmetric about it (7 transitions to its left and 7 transitions to its right). Also, after including transition “5”, the pairing would look like:

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

This inclusion would explain the apparent anomaly. Later, to shift the symmetric point again to {3, 4} the transition “1” is added to the left. Let us take an example of another longest maximal snake in dimension-spread (8, 4) which is of length 19.

Snake (8, 4): 0, 1, 2, 3, 4^{c1}, 5^{c2}, 0^{c3}, 1, 6, 3, 7, 5, 1^{c3}, 2^{c2}, 3^{c1}, 4, 5, 0, 1 (Possible core 1)

Snake (8, 4): 0, 1, 2, 3, 4^{c1}, 5^{c2}, 0^{c3}, 1, 6, 3, 7, 5, 1^{c3}, 2^{c2}, 3^{c1}, 4, 5, 0, 1 (Possible core 2)

The symmetric point in the above snake is transition "3" which is shown in blue color. Since the spread is an even number, the possible two cores encoded in red are listed as its two variants. It also shows all the complementary pairs denoted using the superscripts and are used in the remaining part of the snake. The only complexity the DNA of this snake carries is that there is more than one mapping of the complementary transition pair (e.g. transition "5" is paired with transitions "1" and "2").

3.4 SIMPLE DNA WITH GOOD GENES

In simple DNA, while defining the unique pairing, there are several possible combinations of transition sequences that can be used. These different combinations (similar to gene combinations in the DNA of living cells) decide the length of the longest snake that can be found using the DNA. This important characteristic of being a factor in deciding the length of the longest possible snake, similar to genotype mapping with phenotype in the DNA of living cells, is discussed in this section. Some of these transition combinations that contribute to the longest snakes have been identified.

Consider a snake (11, 5). The core 5 transitions that are used initially have to be different, and can be written as:

3, 1, 0, 2, 4

In this dimension (n = 11), the remaining number of possible transitions to be used in the DNA is even (it is 6). All the longest snakes that were found in [1] belonged to this category of odd dimension and odd spread. After initially placing the first k transitions in a spread-k snake (here k is equal to 5), the next two transitions ("6" and "5") can occur in one of the following ways as shown below:

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

Interestingly, both of these types can contribute to the longest snakes depending on the dimension and spread. In the first type the two transitions outside the core are placed at equal distance from the symmetric point (in the above example, transition "5" is at a distance of three from transition "0" to the left and right side while transition "6" is at a distance of four from transition "0" to the left and right side). This type of DNA contributes to the longest maximal snake in (7, 3).

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

The second one in which the next two transitions form complementary pairs with each other by switching sides on the left and right side of the core, is found in (11, 5). The DNA is shown below:

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

As shown in the above example transition "8" and transition "10" form the complementary pairs. This second type of DNA was also used in the recently found longest snakes in (13, 5), (15, 7) as well as in (17, 7). The first type of DNA, when it was used for (15, 7), grew to be the longest snake of length 57 after an exhaustive search of placing the next complementary pairs. The second type also grew to be the longest snake of length 57. The exhaustive search here refers to the addition of complementary pairs to the left and right side of the DNA, in an exhaustive way as discussed in [1]. The exhaustive search was not complete for (17, 7) and 103 is the length of the longest snake found so far. For the smaller dimension-spread sometimes there are not enough transition options to form these two structures as is the case with snakes in (7, 5). Also for (9, 3) the longest known snake of length 63 was found with the second type of DNA, while the first one could only grow to a snake of length 55. The search was exhaustive for placing the next complementary pairs using both types of DNA in (9, 3). The other good gene combination in the DNA, found in the dimensions searched so far, is to add the complementary transition pairs, nearest to the symmetric point, immediately when it can be added (transition "7" and transition "4" at the second and fourteenth positions respectively in the example below)

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

This is very common in bigger dimensions. For bigger dimensions, there are large numbers of transition pairs that need to be placed on the left and right side of the core structure. Laying all of them simply on alternate sides does not make good DNA for growing long snakes.

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

The DNA shown above is one such example for (11, 5). In this DNA, transition "7" and transition "4" are not placed at the second and fourteenth positions respectively as was done in the previous DNA. This type of DNA does not grow to be the longest maximal snake in (11, 5). The longest snake this DNA can grow is of length 35 (again found using the exhaustive search of placing the complementary pairs), while the longest maximal snake is of length 39.

3.5 RESULTS AND DISCUSSION

While building snakes using the transition sequence and validating these snakes in the transition sequence, it was possible to find some of the longest snakes known so far in dimension 8 through 12 for spread 3, 4 and 5. The results are summarized in Table 3. The values in parentheses indicate the best known results. In Table 3, we see that the exhaustive search was not efficient enough to find the longest snakes in bigger dimensions like dimension 9, 10, 11 and 12 with spread 3. The maximally longest snakes that were found in other dimension-spreads using the exhaustive search were used for DNA analysis and replicating them in other dimension-spreads.

Table 3.1: Canonical Longest Snakes found in dimension-spread

Dimension-Spread	Spread 3	Spread 4	Spread 5
Dimension 8	35(35*)	19(19*)	11(11*)
Dimension 9	58(63)	28(28*)	19(19*)
Dimension 10	-	47(47*)	25(25*)
Dimension 11	-	68(68)	39(39*)
Dimension 12	-	-	55(56)

* indicates the length of the longest maximal snake

For snake (9, 3), a simple DNA analogous to the simple DNA of known maximal snakes of previous dimension-spreads was used to build the known longest snake. Since dimension-spread (9, 3) is an odd dimension with odd spread a simple DNA with unique complementary pair mapping was possible to build. The search space of paired complementary transitions was exhaustively searched for the simple DNA. One of the possible implications, if the DNA approach finds the longest maximal snake, is that the length of the longest maximal snake in (9, 3) is of length 63 since the search was complete using this approach. On the other hand, for snake (11, 3) which also happens to be an odd dimension-odd spread the search space of paired transitions using the DNA was not completed. The best found so far was of length 153. As reported in [1], the other three new longest snakes were also built using this DNA approach. For (15, 7), the search space was exhaustively searched for DNA pairing and the longest found was 57. Since the search was completed, it could also be the longest maximal snake in (15, 7). For (17, 7) and (13, 5), the search was not complete and longer snakes are possible using the complementary pairs. These results are summarized in Table 4.

Table 3.2: Results from the Best Gene Combinations in DNA

Dimension-spread	Good genes
(15, 7)	57[c] (55)
(9, 3)	63[c] (63)
(13, 5)	85 (79)
(17, 7)	103 (98)
(11, 3)	153 (157)

c - Complete search for the given structure

In the previous section, we discussed the various possibilities of placing the transitions inside the DNA and their implications due to varying mapping of complementary pairs. In Table 5, we summarize the results obtained using various gene combinations in the DNA for several dimension-spreads. The second column describes the DNA that was used for searching the snake and the third column describes the longest snake that was found using the DNA. For some of the dimension-spreads the search was complete (length is marked by a superscript c) while for others longer snakes are possible. The values in parentheses in the third column are the length of the previously known longest snakes in the dimension-spread. An asterisk * means that it is the length of the longest maximal snake in the dimension-spread.

In this paper we demonstrated the possibility of building snakes in dimension and spread combinations other than the odd dimension odd spread. The DNA for these remaining combinations is complex and uses multiple mapping. The usefulness of this type of DNA is not much appreciated because there was no new records established using this type of DNA. This complex DNA is able to explain various longest snakes while for few others, it cannot completely explain using a single hypothesis. For the Simple DNA, the results obtained were astonishing and we were able to break three new records while tying with another. Simple DNA which is found in odd dimension with odd spread shows promising results and can be used to explore further. One of the important area for exploration could be to predict the next complementary pairs outside the DNA that could help in growing the longest snake.

Table 3.3: Longest Snakes Built Using Various Gene Combinations in the DNA

Dimension-spread	DNA	Length
(11, 5)	**9, 2, 7, 6, 5, 3, 1, 0, 2, 4, 6, 5, 8, 1, 10**	**39^c(39*)**
(11, 5)	9, 7, 6, 5, 3, 1, 0, 2, 4, 6, 5, 8, 10	35^c(39*)
(11, 5)	9, 2, 7, 6, 5, 3, 1, 0, 2, 4, 5, 6, 8, 1, 10	33^c(39*)
(9, 3)	**7, 5, 4, 3, 1, 0, 2, 4, 3, 6, 8**	**63^c(63)**
(9, 3)	7, 2, 5, 4, 3, 1, 0, 2, 4, 3, 6, 1, 8	57^c(63)
(9, 3)	7, 2, 5, 4, 3, 1, 0, 2, 3, 4, 6, 1, 8	57^c(63)
(9, 3)	7, 5, 4, 3, 1, 0, 2, 3, 4, 6, 8	55^c(63)
(15, 7)	**13, 11, 2, 9, 8, 7, 5, 3, 1, 0, 2, 4, 6, 8, 7, 10, 1, 12, 14**	**57^c(55)**
(15, 7)	**13, 4, 11, 2, 9, 8, 7, 5, 3, 1, 0, 2, 4, 6, 8, 7, 10, 1, 12, 3, 14**	**57^c(55)**
(15, 7)	**13, 11, 2, 9, 8, 7, 5, 3, 1, 0, 2, 4, 6, 7, 8, 10, 1, 12, 14**	**57^c(55)**
(15, 7)	13, 11, 9, 8, 7, 5, 3, 1, 0, 2, 4, 6, 8, 7, 10, 12, 14	53^c(55)
(15, 7)	13, 11, 9, 8, 7, 5, 3, 1, 0, 2, 4, 6, 7, 8, 10, 12, 14	51^c(55)
(15, 7)	13, 11, 9, 7, 5, 3, 1, 0, 2, 4, 6, 8, 10, 12, 14	45^c(55)
(13, 5)	**11, 9, 2, 7, 6, 5, 3, 1, 0, 2, 4, 6, 5, 8, 1, 10, 12**	**85(79)**
(13, 5)	**11, 9, 7, 6, 5, 3, 1, 0, 2, 4, 6, 5, 8, 10, 12**	**85(79)**
(17, 7)	**15, 13, 11, 2, 9, 8, 7, 5, 3, 1, 0, 2, 4, 6, 8, 7, 10, 1, 12, 14, 16**	**103(98)**
(17, 7)	15, 13, 4, 11, 2, 9, 8, 7, 5, 3, 1,0,2, 4, 6, 8, 7, 10, 1, 12, 3, 14, 16	93(98)
(11, 3)	9, 7, 5, 4, 3, 1, 0, 2, 4, 3, 6, 8, 10	153(157)

c - Complete search for the given DNA

* indicates the length of the longest maximal snake

REFERENCES

[1] Khan, Md. S. and Potter, W. D. (2015). “The DNA of Snakes”, submitted to The International Conference on Artificial Intelligence’15.

[2] Meyerson, S., Whiteside, W., Drapela, T., and Potter, W.D. (2014). “Finding Longest Paths in Hypercubes: Snakes and Coils,” *in Proceedings of the IEEE Symposium on Computational Intelligence for Engineering Solutions, IEEE SSCI' 2014, Orlando, FL* (to appear)

[3] Meyerson, S., Whiteside, W., Drapela, T., and Potter, W. D. (2014). “Finding Longest Paths in Hypercubes, 11 New Lower Bounds: Snake, Coils, and Symmetric Coils,” (under review).

[4] G. Zémor. (1997). “An upper bound on the size of the snake-in-the-box”, *Combinatorica*, 17, 287–298

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

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

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

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

[9] D. A. Casella and W. D. Potter. (2004). “New lower bounds for the snake-in-the-box problem: Using evolutionary techniques to hunt for snakes”, *In Proceedings of the Florida Artificial Intelligence Research Society Conference*, pages 264-268

[10] 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

CHAPTER 4

SNAKE REPRESENTATIONS

Snakes can be represented in various ways. Here we discuss some of the existing representations of snakes and a possible new representation based on the ideas discussed in the thesis. Representations of snake can play an important role in optimizing any search technique with respect to its time and space complexity. Some of the few common search optimizations may include pruning the search space and reducing the time required to validate snakes.

4.1 OVERVIEW

Two of the most common representations that are used in the current literature are node-sequence representation and transition sequence representation. In node-sequence representation we use the nodes of an n-dimensional hypercube that are traversed along the path as we move in the hypercube. Transition sequence representation on the other hand represents the change in bit position when one moves from one node to the other. While counting the bit position, one can start with either 0 as the first position or 1 as the first position. Based on this we call it a 0-based or 1-based transition sequence. Both the representations have their own advantages and disadvantages. Node sequence representation is one of the simplest representations but the number of nodes increases exponentially as we move to higher dimensions. To validate snakes in this representation one usually stores a map of neighboring nodes for each node or converts each node into its bit representation and uses bit operations to find the distance between consecutive nodes.

On the other hand the transition sequence representation uses less terms to represent a snake in an n-dimensional hypercube as it uses the change in bit position while moving from one node to another and discards the concept of absolute node. The number of transitions increases linearly as we move to higher dimensions. This can play a very important role for very large dimensions. Validating a transition sequence gets tricky and in this thesis we introduced the first ever algorithm which validates the transition sequence in its native form without converting it into any other representation.

4.2 FIBONACCI SNAKE REPRESENTATION

In this article we discuss a new representation of a snake for the Snake in the Box problem. This representation facilitates easier validation of snakes. The proposed representation is grounded in the basic idea of differentiating all the distinct transitions in an n-dimensional hypercube. This representation is called the Fibonacci Snake Representation, because, similar to a Fibonacci series where the next term is the sum of previous terms, this representation also uses the sum of previous terms to determine the next term. This representation has its range in natural numbers (excluding 0) which it uses in selecting its terms for representing distinct transitions in an n-dimensional hypercube.

4.2.1 Terms in FSR

A snake transition sequence 0, 1, 2, 3, 1, 0 can be represented as f_1, f_2, f_3, f_4, f_2, f_1 where f_i is the ith term in this Fibonacci Snake Representation (FSR). For choosing f_{i+1} the constraint is that the sum of all the terms up till f_i should be less than f_{i+1}, i.e. $f_1 + f_2 +..+ f_i < f_{i+1}$. Also, when re-using the terms in a sequence, the sign of the term alternates between + and -. So in fact the above sequence would exactly look like $+f_1$, $+f_2$, $+f_3$, $+f_4$, $-f_2$, $-f_1$. It says that 0 in the transition sequence would be represented by some term $+f_1$ in FSR. Likewise 1 as $+f_2$, 2 as $+f_3$, 3 as $+f_4$, next 1 as $-f_2$ and next 0 as $-f_1$.

One such FSR would be all the terms from the set of numbers containing the powers of 2. We can call it Binary FSR (BFSR is a special case of FSR). In BFSR, $f_1 = 1$, $f_2 = 2$, $f_3 = 4$, $f_4 = 8$ and so on. It satisfies the constraint $f_1 + f_2 +..+ f_{i-1} < f_i$. For the BFSR, it satisfies the constraint $f_1 + f_2 +..+ f_i < f_{i+1}$ since $f_1 < f_2$ (as $1 < 2$), $f_1 + f_2 < f_3$ (as $1 + 2 < 4$). It also satisfies that $f_1 + f_2 + f_3 < f_4$ (as $4+2+1 < 8$).

4.2.2 Validation

To validate a snake in FSR, we need to sum all the terms in the sequence. The resultant sum should be the sum of at least k terms to satisfy a spread k snake. While summing the sequence, the first occurrence of any term should be positive. If the first occurrence of any term is not positive then while validating the sequence this term is multiplied with -1 in each occurrence before adding it. So say for example we are validating a part of the snake which is -1, -4, 0, 8, 1. To validate this which could be a part of a longer snake we change it to 1, 4, 0, 8, -1 by changing the sign of -1 and -4 such that their first occurrence is positive. For sequences whose sum is zero, they do not satisfy any spread snake and are invalid snake sequences however some subsequence may be valid. In short, the sum would determine the maximum spread the sequence can satisfy as a valid snake.

Let us explain this with a BFSR example. So for the previous transition sequence (sequence 0, 1, 2, 3, 1, 0) the corresponding BFSR is 1, 2, 4, 8, -2, -1. We get this by using the numbers that are exact powers of the natural numbers. So 2^0 is 1 and 2^1 is 2 and so on. These numbers satisfy the FSR constraint. By summing the complete sequence we get 12 which is the sum of 2 terms (8 + 4) and can be a valid spread 2 snake (number of terms define the spread). This is very useful for validating lower spread snakes since for invalid snakes the sum would correspond either to an individual term or sum of fewer terms. This is because the number of terms to obtain the sum describes the maximum spread for which the sequence can be valid. For a spread 2 snake adding the term -8 to the sequence (1, 2, 4, 8, -2, -1) would make it an invalid spread 2 snake as the sum decreases to 4 and is not a sum of at least 2 terms. Also while validating, the difference between sums at each stage should be the sum of at least k terms in a valid spread k snake. So 1, 2, 4, 8, -2 has an incremental sum of 1, 3, 7, 15, 13. When we add 1 with 2 we get 3 and when we add 4 to it we get 7 and so on and so forth we get this incremental sum. For these incremental sum we see that the first sum 1 and the fifth sum 13 have a difference of 12 (which can be expressed as a sum of two terms 4 and 8) and thus cannot be a valid snake for spread 3. These sums at each stage do not change while building snakes and thus need not be re-generated with each addition of a new transition.

CHAPTER 5

CONCLUSION

In this thesis, we reported the discovery of a general underlying pattern among many of the longest maximal snakes. These underlying patterns - the DNA of snakes, were found to be very similar to the DNA found in cells of living organisms. Extending this idea of DNA, snakes were built using these underlying structures. These structures successfully grew into the longest snakes in as many as four dimension-spreads whose longest maximal are yet to be known (among these four, three of them are the new known longest snakes). This approach brought an NP-Hard problem one step closer to being generalized. The approach being general can be used to hunt for snakes in dimension-spreads which have not been tried yet. As this approach of using DNA with predefined base-pairing reduces the search space to a much smaller search space, it helps in easing the computationally intensive search which otherwise with current computing resources is too difficult to search. One of the limitations of this algorithm is that for complex DNA the mapping is not-unique and using it is not a trivial task. In fact for complex DNA, the empirical evidence is not sufficient enough to build a solid hypothesis and is considered as an area of possible future work for research. For snakes with simple DNA, the search of placing the next complementary pair is a simple exhaustive search. As we go in higher dimensions, another possible research direction would be to find a way to predict the next complementary transition pair while building the snake from its DNA.

BIBLIOGRAPHY

Abbott, H. L. and M. Katchalski. (1991). “On the Construction of Snake-In-The-Box Codes”. *Utilitas Mathematica* 40:97-116

Harary, F., J. P. Hayes, and H. J. Wu. 1988. “A Survey of the Theory of Hypercube Graphs”. *Computational Mathematics Applications* 15:277-289

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

S. Hood, J. Sawada, C.H. Wong, (2010). “Generalized Snakes and Coils in the Box”

Kautz. W. H. (1958). “Unit-Distance Error-Checking Codes”. *IRE Trans. Electronic Computers* 7:179-180

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

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

Y. Yehezkeally, M. Schwartz. (2012). “Snake-in-the-box codes for rank modulation”
IEEE Trans. Inform. Theory, 58, 5471–5483

APPENDIX A

THE LONGEST SNAKES FOR NEW LOWER BOUNDS

A.1.1 SNAKE (13, 5) - LENGTH 85

[0, 1, 2, 3, 4, 5, 6, 1, 7, 3, 8, 2, 9, 10, 1, 6, 7, 11, 4, 12, 9, 3, 10, 1, 6, 0, 2, 3, 8, 1, 7, 10, 12, 2, 3, 11, 5, 8, 7, 4, 12, 2, 3, 6, 1, 7, 4, 0, 9, 10, 3, 6, 1, 11, 4, 12, 0, 3, 6, 8, 2, 12, 11, 3, 5, 1, 7, 10, 4, 2, 12, 11, 5, 6, 0, 3, 4, 12, 2, 9, 7, 3, 6, 12, 8]

A.1.2 SNAKE (13, 5) - LENGTH 85

[0, 1, 2, 3, 4, 5, 6, 7, 1, 3, 8, 5, 9, 6, 0, 10, 8, 3, 11, 2, 0, 9, 12, 8, 3, 4, 5, 2, 7, 11, 9, 3, 4, 6, 0, 2, 5, 1, 10, 7, 9, 4, 6, 5, 12, 10, 7, 11, 4, 3, 8, 6, 5, 2, 12, 1, 10, 3, 4, 5, 2, 0, 9, 12, 8, 3, 1, 2, 0, 7, 8, 6, 12, 4, 0, 2, 11, 10, 6, 4, 5, 2, 3, 11, 8]

A.2 SNAKE (15, 7) - LENGTH 57

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 3, 0, 1, 7, 12, 8, 4, 13, 11, 0, 14, 10, 7, 5, 8, 6, 4, 0, 9, 1, 10, 7, 2, 6, 12, 3, 4, 9, 11, 10, 5, 6, 13, 12, 7, 8, 9, 10, 0, 1, 4, 13, 14, 5, 6]

A.3 SNAKE (17, 7) - LENGTH 103

[0, 1, 2, 3, 4, 5, 6, 7, 8, 0, 9, 2, 4, 10, 11, 3, 8, 12, 7, 13, 2, 6, 14, 3, 9, 1, 7, 12, 4, 2, 11, 15, 3, 8, 1, 5, 4, 12, 0, 6, 15, 2, 16, 7, 4, 9, 13, 8, 1, 12, 0, 15, 4, 5, 10, 13, 8, 3, 0, 14, 11, 6, 15, 2, 4, 5, 0, 12, 10, 13, 9, 15, 16, 6, 0, 5, 14, 10, 3, 9, 7, 2, 6, 8, 14, 5, 13, 9, 16, 1, 0, 6, 3, 4, 13, 14, 2, 12, 0, 9, 6, 10, 4]

APPENDIX B

THE LONGEST CANONICAL SNAKES

B.1 SNAKES WITH SPREAD 2

(6, 2) - 26 - [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]

(7, 2) - 50 - [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]

B.2 SNAKES WITH SPREAD 3

(6, 3) - 13 - [0, 1, 2, 3, 0, 4, 2, 5, 0, 1, 2, 3, 0]

(7, 3) - 21 - [0, 1, 2, 3, 0, 4, 5, 1, 0, 3, 6, 4, 0, 1, 2, 3, 0, 4, 5, 1, 0]

(8, 3) - 35 - [0, 1, 2, 3, 0, 4, 2, 5, 6, 1, 4, 2, 7, 3, 6, 1, 4, 0, 6, 7, 1, 5, 3, 7, 6, 2, 5, 4, 1, 6, 5, 0, 3, 4, 1]

(9, 3) - 63 - [0, 1, 2, 3, 0, 4, 2, 5, 0, 1, 6, 7, 5, 4, 1, 0, 3, 4, 2, 0, 5, 4, 1, 7, 0, 4, 6, 7, 3, 4, 2, 8, 0, 3, 4, 1, 5, 3, 2, 1, 7, 3, 6, 2, 0, 3, 4, 2, 7, 3, 6, 1, 5, 7, 2, 6, 0, 3, 2, 4, 0, 7, 2]

B.3 SNAKES WITH SPREAD 4

(7, 4) - 11 - [0, 1, 2, 3, 4, 0, 5, 1, 6, 2, 0]

(8, 4) - 19 - [0, 1, 2, 3, 4, 5, 0, 1, 6, 3, 7, 5, 1, 2, 3, 4, 5, 0, 1]

[0, 1, 2, 3, 4, 5, 0, 2, 6, 3, 7, 5, 2, 1, 3, 4, 5, 0, 2]

(9, 4) - 28 - [0, 1, 2, 3, 4, 0, 5, 1, 6, 7, 2, 0, 1, 4, 8, 7, 0, 3, 4, 5, 7, 2, 0, 1, 6, 5, 4, 0]

(10, 4) - 47 - [0, 1, 2, 3, 4, 0, 5, 6, 3, 1, 7, 0, 5, 8, 1, 2, 3, 5, 4, 6, 1, 0, 2, 5, 9, 1, 6, 8, 3, 2, 5, 6, 4, 8, 7, 0, 2, 1, 4, 6, 3, 8, 2, 4, 5, 0, 8]

(11, 4) - 68 - [0, 1, 2, 3, 4, 0, 5, 1, 6, 2, 0, 7, 1, 3, 2, 8, 0, 4, 1, 2, 9, 0, 5, 1, 3, 7, 0, 2, 1, 6, 7, 4, 3, 8, 0, 10, 5, 4, 7, 6, 1, 0, 4, 5, 2, 1, 8, 9, 0, 2, 3, 7, 4, 8, 0, 2, 1, 6, 9, 5, 4, 8, 0, 7, 9, 4, 1, 0]

B.4 SNAKES WITH SPREAD 5

(8, 5) - 11 - [0, 1, 2, 3, 4, 5, 0, 6, 1, 7, 2]

(9, 5) - 19 - [0, 1, 2, 3, 4, 5, 0, 6, 2, 7, 4, 8, 0, 1, 2, 3, 4, 5, 0]

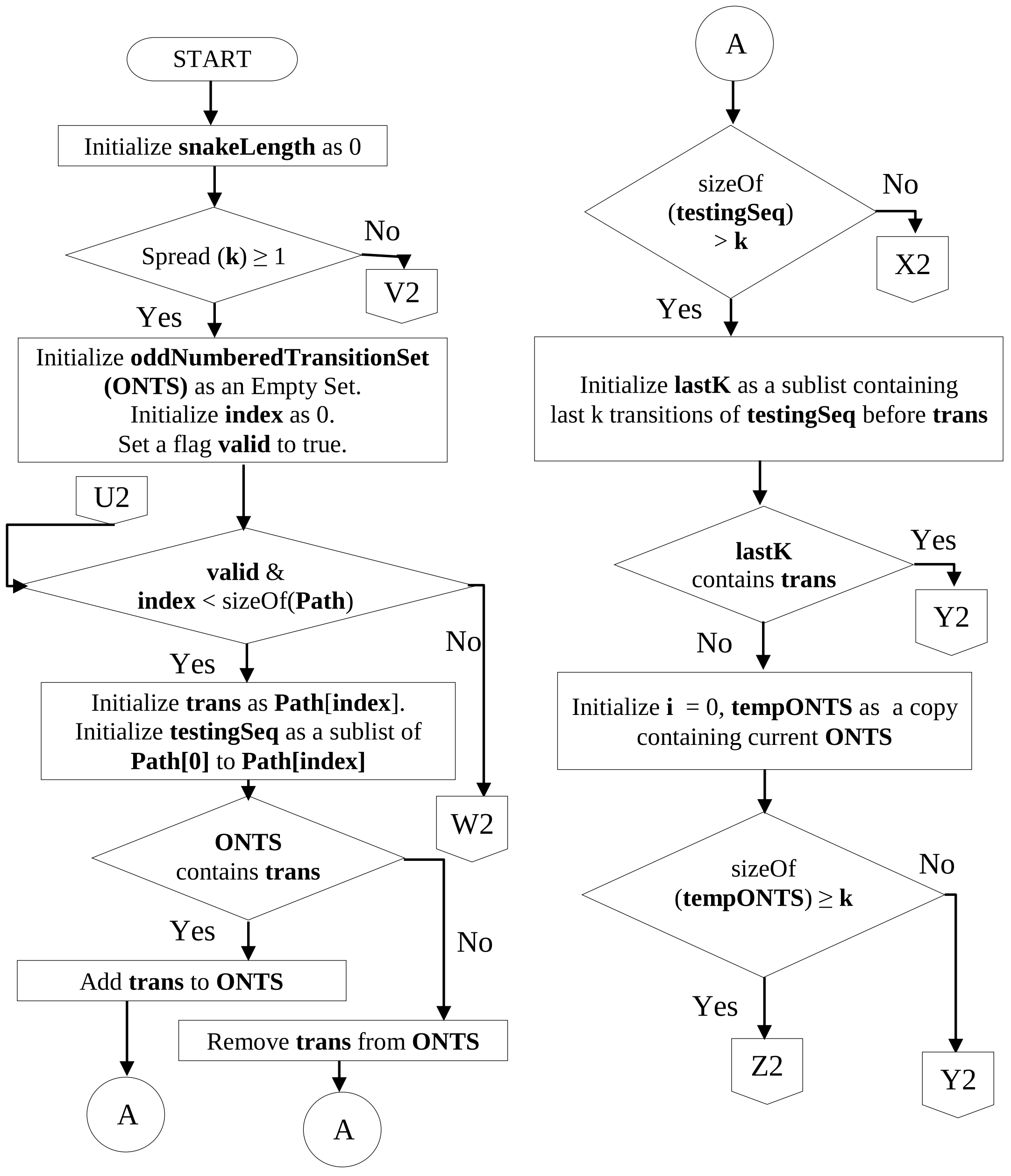
(10, 5) - 25 - [0, 1, 2, 3, 4, 5, 0, 6, 2, 7, 3, 8, 5, 9, 2, 1, 7, 0, 5, 4, 3, 2, 7, 6, 0]

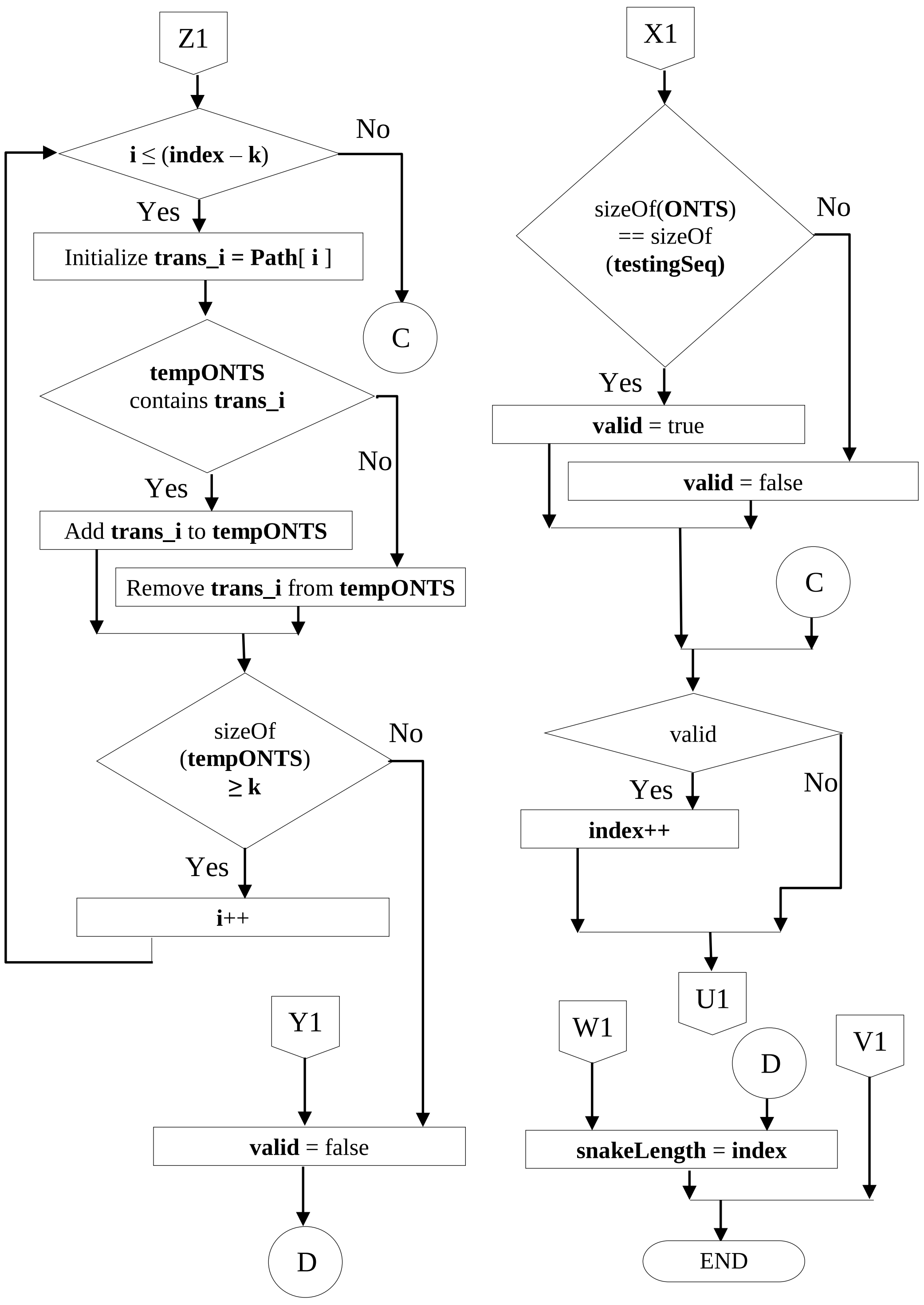
(11, 5) - 39 - [0, 1, 2, 3, 4, 5, 6, 7, 1, 2, 8, 5, 9, 7, 0, 10, 8, 1, 4, 5, 7, 6, 10, 8, 3, 4, 2, 5, 10, 9, 6, 4, 1, 5, 7, 0, 9, 6, 3]

APPENDIX C

VALIDATING SNAKE IN TRANSITION SEQUENCE

C.1 FLOWCHART


START
Initialize **snakeLength** as 0
Spread (**k**) ≥ 1
No
V2
Yes
Initialize **oddNumberedTransitionSet (ONTS)** as an Empty Set. Initialize **index** as 0. Set a flag **valid** to true.
U2
valid & **index** < sizeOf(**Path**)
No
Yes
Initialize **trans** as **Path**[**index**]. Initialize **testingSeq** as a sublist of **Path[0]** to **Path[index]**
W2
ONTS contains **trans**
Yes
No
Add **trans** to **ONTS**
Remove **trans** from **ONTS**
A
A
A
sizeOf (**testingSeq**) > **k**
No
X2
Yes
Initialize **lastK** as a sublist containing last k transitions of **testingSeq** before **trans**
lastK contains **trans**
Yes
Y2
No
Initialize **i** = 0, **tempONTS** as a copy containing current **ONTS**
sizeOf (**tempONTS**) ≥ **k**
No
Yes
Z2
Y2



Z1
i ≤ (index – k)
No
Yes
Initialize trans_i = Path[i]
C
tempONTS contains trans_i
No
Yes
Add trans_i to tempONTS
Remove trans_i from tempONTS
sizeOf (tempONTS) ≥ k
No
Yes
i++
Y1
valid = false
D
X1
sizeOf(ONTS) == sizeOf (testingSeq)
No
Yes
valid = true
valid = false
C
valid
Yes
No
index++
U1
W1
D
V1
snakeLength = index
END