FINDING LONGEST PATHS IN HYPERCUBES: 11 NEW LOWER BOUNDS FOR SNAKES, COILS, AND SYMMETRICAL COILS

by

SETH MEYERSON

(Under the Direction of Walter D. Potter)

ABSTRACT

Abstract— Since the problem's formulation by Kautz in 1958 as an error detection tool, diverse applications for long snakes and coils have been found. These include coding theory, electrical engineering, and genetics. Over the years, the problem has been explored by many researchers in different fields using varied approaches, and has taken on additional meaning. The problem has become a benchmark for evaluating search techniques in combinatorially expansive search spaces (NP-complete Optimizations).

In two papers, the second building upon the first, we present an effective process for searching longest induced paths: open (snakes), closed (coils), and symmetric closed (symmetric coils) in n-dimensional hypercube graphs. Stochastic Beam Search, a non-deterministic variant of Beam Search, provides the overall structure for our search, while graph theory based techniques are used in the computation of a generational fitness value. This novel fitness value is used in guiding the search. We present eleven new lower bounds for the Snake-in-the-Box problem for snakes in dimensions 11, 12, and 13; coils in dimensions 10, 11, and 12; and symmetric coils in dimensions 9, 10, 11, 12, and 13. The best known solutions of the unsolved

dimensions of this problem have improved over the years and we are proud to make a contribution to this problem as well as the continued progress in combinatorial search techniques.

INDEX WORDS: stochastic beam search; snake-in-the-box; combinatorial optimization; graph search; hypercube; heuristic search

FINDING LONGEST PATHS IN HYPERCUBES: 11 NEW LOWER BOUNDS FOR SNAKES, COILS, AND SYMMETRICAL COILS

by

SETH MEYERSON

BS, Appalachian State University, 1996

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

2015

FINDING LONGEST PATHS IN HYPERCUBES: 11 NEW LOWER BOUNDS FOR SNAKES, COILS, AND SYMMETRICAL COILS

by

SETH MEYERSON

Major Professor:	Walter D. Potter
Committee:	Krzysztof J. Kochut
	Khaled Rasheed

Electronic Version Approved:

Julie Coffield
Interim Dean of the Graduate School
The University of Georgia
May 2015

DEDICATION

To my family – Catharine, Elisabeth, Ben, Sam, Dad and Debbie

ACKNOWLEDGEMENTS

I thank Dr. Walter Potter for introducing me to the amazing world of computational intelligence, and challenging me to work on and supporting my efforts to set new lower bounds for the well-studied problem at the center of this document. I thank Dr. Khaled Rasheed and Dr. Lakshmish Ramaswamy for their patience, guidance, and encouragement, both with the problem and the University of Georgia's Computer Science Master's program. I thank friend and colleague Tom Drapela for his invaluable help with this project- from white board, to the IDE, and as an editor and contributor to the two published papers contained in this thesis. Thank you to Will Whiteside for the idea of applying graph algorithms to the problem. I thank Dustin Cline and Thomas Horton for their contributions to this project, and the Institute for Artificial Intelligence.

This study was partially supported by resources and technical expertise from the University of Georgia Computer Science Department. Particularly Ken for "have you thought about doing this in parallel?" This study was also partially supported by resources and technical expertise from the Georgia Advanced Computing Resource Center (GACRC), a partnership between the University of Georgia's Office of the Vice President for Research and Office of the Vice President for Information Technology.

I thank friends Tom Werth, and Seamus Narron, who supported and shared my journey at the University of Georgia.

Finally, I thank my beautiful wife Catharine for her unwavering love and support as I made my way through the University of Georgia's Computer Science Master's program, including this thesis – I could not have done it without her.

TABLE OF CONTENTS

Page

CHAPTER 1

INTRODUCTION AND LITERATURE REVIEW

The Snake-in-the-Box problem (SIB) is a computationally hard problem concerned with finding longest induced (achordal) paths through an n-dimensional hypercube (or n-cube) graph. Finding longest induced paths is well known to be NP-complete (Garey, 1979). A hypercube graph consists of a set of binary numbered nodes {0…2n-1} in which two nodes are adjacent if and only if their binary values differ by exactly one bit.

An achordal path is a path in which each node in the path is only adjacent to its immediate predecessor and successor nodes within the path. In the SIB problem, open achordal paths are called snakes, and closed achordal paths are called coils. For snakes, any path which cannot be extended is called a maximal path, with the longest of these being the longest maximal path. The longest maximal snake for any n-cube is the snake solution (absolute bound) for SIB in dimension-n. Similarly, the longest possible coil for any n-cube is the coil solution (absolute bound) for dimension-n. For these and other SIB terms and definitions, please visit the UGA Institute for Artificial Intelligence's webpage (Potter, 2014).

Introduced by Kautz in 1958, the Snake-in-the-Box problem (SIB) has been extensively studied for over 50 years. The problem has found relevance in many fields including coding theory (Kautz, 1958), electrical engineering (Klee, 1970), analog to digital conversion (Hiltgen, 2000), precision high speed rotational measurement devices (Zhang, 2008), disk sector encoding

(Blaum, 2001), rank modulation schemes for charge distribution in multi-level flash memories (Yehezkeally, 2012), and genetics related to embryonic development (Zinovik, 2010).

In addition the problem has become a benchmarking tool for combinatorial search. Contributions have been made by experts in diverse fields such as computer science, mathematicians, artificial intelligence, social informatics, and chemical engineering. The approaches to the problem have been just as diverse.

For dimensions $n < 7$, snake and coil solutions were computed via exhaustive search by Davies (Davies, 1965). Solutions for $n = 7$ were determined via genetic algorithm by Potter (Potter, 1994) and later verified by the exhaustive techniques of Kochut (Kochut, 1996) which we will be building upon.

For dimensions $n > 7$, exhaustive search has not been feasible for finding solutions due to the exponential growth of the problem as dimensionality increases. It is estimated that it would take many years of computing time to exhaustively search dimension 8 (Kinny, 2012).

In lieu of this, non-exhaustive techniques, both computational search and mathematical construction based approaches, have been applied to improving the known lower bounds of higher dimensions. These include: genetic algorithms (Potter, 1994), hybrid genetic algorithms (Carlson, 2010), nested Monte Carlo Search schemes (Kinny, 2012), particle swarm optimization (Brooks, 2012), population-based stochastic hill-climbers (Casella, 2005), distributed heuristic computing (Juric, 1994), neural networks (Bishop, 2006), hybrid evolutionary algorithms (Casella, 2005) and mathematical constructive techniques (Adelson, 1973)(Abbott, 1991)(Wynn, 2014). With technical improvement in searches, and improvements in computing power, the lower bounds for these higher dimensions have inevitably improved. We are pleased to make our contribution to the SIB problem and combinatorial search.

Chapter 2 contains paper titled “Finding Longest Paths in Hypercubes, snakes and coils” published in the proceedings of the IEEE Symposium on Computational Intelligence for Engineering Solutions, CIES ’14 held in Orlando, FL. In Chapter 2 we apply a new search technique (new to SIB) along with a novel approach to evaluating the fitness of sub solutions through graph based searches of the remaining hypercube space. This results in several new lower bounds in dimensions 10-12 of the problem.

Chapter 3 contains a paper titled “Finding Longest Paths in Hypercubes, 11 New Lower Bounds: Snakes, Coils, and Symmetric Coils.” which will appear in the proceedings of The 28th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, (IEA/AIE’15). In chapter 3 we build upon on our work in chapter 2. We develop new graph theory based evaluations of the hypercube state resultant from sub-solutions in order to improve the predictive fitness value used to prune the inevitable combinatorial explosion. In chapter 3 we also develop a related approach for searching for longest symmetrical coil solutions. Our efforts yield eleven new record lower bounds for the problem; the records in dimensions 9-13, encompassing snakes, coils, and symmetrical coils.

In chapter 4 we conclude with a summary and few observations and predictions about the problem and a tip of our hat to Abbott.

CHAPTER 2

FINDING LONGEST PATHES IN HYPERCUBES, SNAKES AND COILS[1]

[1] Meyerson, S., Whiteside, W., Drapela, T., and Potter, W.D., "Finding Longest Paths in Hypercubes: Snakes and Coils,"in *Proceedings of the IEEE Symposium on Computational Intelligence for Engineering Solutions*, IEEE CIES'14, pp. 103-109, Orlando, FL, December, 2014 – Reprinted here with permission of publisher

Abstract

Since the problem's formulation by Kautz in 1958 as an error detection tool, diverse applications for long snakes and coils have been found. These include coding theory, electrical engineering, and genetics. Over the years, the problem has been explored by many researchers in different fields using varied approaches, and has taken on additional meaning. The problem has become a benchmark for evaluating search techniques in combinatorially expansive search spaces (NP-complete Optimizations).

We present an effective process for searching for long achordal open paths (snakes) and achordal closed paths (coils) in n-dimensional hypercube graphs. Stochastic Beam Search provides the overall structure for the search while graph theory based techniques are used in the computation of a generational fitness value. This novel fitness value is used in guiding the search. We show that our approach is likely to work in all dimensions of the SIB problem and we present new lower bounds for a snake in dimension 11 and coils in dimensions 10, 11, and 12. The best known solutions of the unsolved dimensions of this problem have improved over the years and we are proud to make a contribution to this problem as well as the continued progress in combinatorial search techniques.

1. Introduction

The Snake-in-the-Box problem (SIB) is a computationally hard problem concerned with finding longest induced (achordal) paths through an n-dimensional hypercube (or n-cube) graph. Finding longest induced paths is well known to be NP-complete [1]. A hypercube graph consists of a set of binary numbered nodes $\{0\ldots2^n\text{-}1\}$ in which two nodes are adjacent if and only if their binary values differ by exactly one bit. Fig. 1 illustrates a 3-dimensional hypercube.

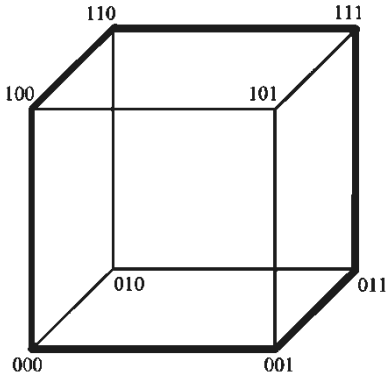

Figure 1: 3-dimensional hypercube (with embedded coil in bold).

An achordal path is a path in which each node in the path is only adjacent to its immediate predecessor and successor nodes within the path. In the SIB problem, open achordal paths are called snakes, and closed achordal paths are called coils. For snakes, any path which cannot be extended is called a maximal path, with the longest of these being the longest maximal path. The longest maximal snake for any n-cube is the snake solution (absolute bound) for SIB in dimension-n. Similarly, the longest possible coil for any n-cube is the coil solution (absolute bound) for dimension-n. For these and other SIB terms and definitions, please visit the UGA Institute for Artificial Intelligence's webpage [2].

Introduced by Kautz in 1958, the Snake-in-the-Box problem (SIB) has been extensively studied for over 50 years. The problem has found relevance in many fields including coding theory [3], electrical engineering [4], analog to digital conversion [5], precision high speed rotational measurement devices [6], disk sector encoding [7], rank modulation schemes for charge distribution in multi-level flash memories [8], and genetics related to embryonic development [9].

For dimensions n < 7, snake and coil solutions were computed via exhaustive search by Davies [10]. Solutions for n = 7 were determined via genetic algorithm by Potter [11] and later verified by the exhaustive techniques of Kochut [12] which we will be building upon.

For dimensions n > 7, exhaustive search has not been feasible for finding solutions due to the exponential growth of the problem as dimensionality increases. It is estimated that it would take many years of computing time to exhaustively search dimension 8 [13]. In lieu of this, non-exhaustive techniques, both computational search and mathematical construction based approaches, have been applied to improving the known lower bounds of higher dimensions. These include: genetic algorithms [11], hybrid genetic algorithms [14], nested Monte Carlo Search schemes [15], particle swarm optimization [16], population-based stochastic hill-climbers [17], distributed heuristic computing [18], neural networks [19], hybrid evolutionary algorithms [17] and mathematical constructive techniques [20] [21] [22]. With each successive technical improvement, enhanced by increases in computing power, the lower bounds for these higher dimensions continue to be improved. Table I shows the solution lengths of n-snakes and n-coils for n < 7. [23].

TABLE 1: Absolute Bounds

Dimension	**Length**	
	Snakes	***Coils***
1	1	0
2	2	4
3	4	6
4	7	8
5	13	14
6	26	26
7	50	48

2 Stochastic Beam Search Overview

We apply a new search method, Beam Search (new to SIB): a non-deterministic search, combining elements of graph search with techniques common to evolutionary computing. Beam search is a heuristic search algorithm for combinatorial optimization problems which has been extensively studied in artificial intelligence and operations research [24]. "An optimization of the best first search graph search algorithm where only a predetermined number of paths are kept as candidates. The number of paths is the "width of the beam". If more paths than this are generated, the worst paths are discarded. This reduces the space requirements of best first search." [25] The term "beam search" was coined by Raj Reddy, at Carnegie Mellon University in 1976. Beam Search is related to breadth-first search and is also similar to best-first search. Our search progresses by level through a structured snake or coil search tree containing all possible valid sub-paths, called sub-solutions. With each iteration of the search, a new level of the search tree is explored. At each level every newly encountered valid node is added to the path. Each unique node addition to a sub-solution requires an addition to the next new population of sub-solutions. Each sub-solution in the new population is evaluated using a heuristic fitness function, the less fit solutions (branches) are discarded (pruned) in order to reduce the number of sub-solutions to a fixed limit; the beam width. Like many heuristic approaches, beam search cannot guarantee that the optimal solution will be found, but properly applied it can give good or even excellent results in exponentially expansive search spaces [26].

Fig. 2 shows beam search through successive expansion of selected solutions. The 'hypercubes' within the circles represent the candidate solutions that have been selected for continued exploration in the next iteration.

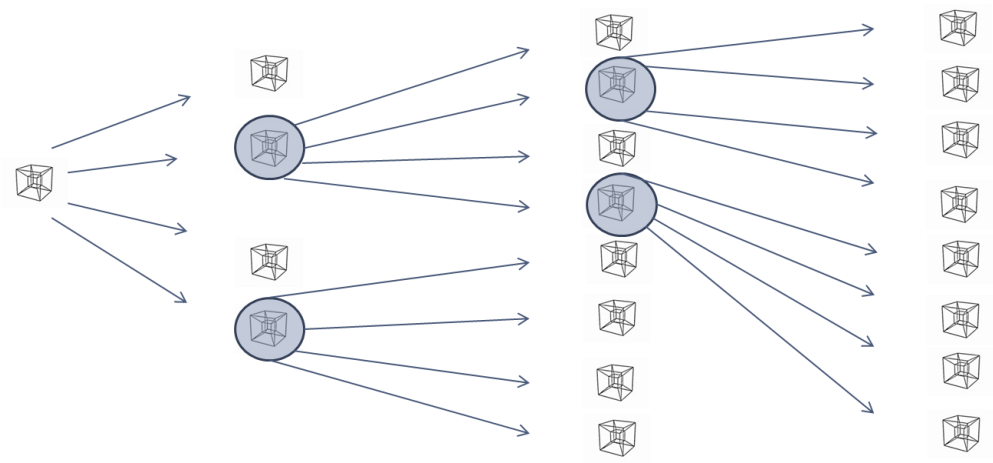

Figure 2: Beam Search

The non-selected (un-circled) solutions are discarded. The beam width in this case is 2. Notice there is no selection while the number of leaf nodes is less than the beam width.

Stochastic Beam Search differs from a Beam Search only in how the decision to prune is made. A standard implementation of beam search would rank and retain only the fittest sub-solutions that fit within the beam width. Stochastic Beam Search is also guided by fitness, but provides for randomness similar to the tournament selection process commonly used in Genetic Algorithms (GA). Stochastic selection will be described in more detail later.

We search for long snakes and coils separately taking advantage of some specific optimizations that allow for a better coil search. A summary of our implementation of Stochastic Beam Search with tailored fitness metrics for the SIB problem is described in the following steps.

2.1 Initialization

When searching for both snakes and coils, we begin with a population of one candidate sub-solution, the valid snake described by the node sequence {0, 1, 3, 7}. At this length any other valid sequence is merely an isomorphic duplication of this sequence. Like many others [12],

[11], [15], [14] we begin our search with node 0. See the UGA AI website for definitions of Canonical form [2]. There is no need for any other solution in the initial population as any other sub-solutions at this length would entail isomorphic redundancy.

2.2 Construction Step

We construct a new population of sub-solutions from the previous population of sub-solutions by extending the length of each sub-solution in the old population with the addition of a valid next node. If there is more than one valid extension for a sub-solution in the old population, a new sub-solution is created in the new population for every valid extension. Thus each new sub-solution is created from a parent sub-solution and differs from other sub-solutions with the same parent only by its newly appended node. This iterative scheme generates successive populations / levels of snake sub-solutions that are exactly one node longer than the sub-solutions of the previous population. The previous population is discarded. If no valid next node is available for a sub-solution, then this sub-solution is terminal and no descendant of it will appear in the new population. If a valid next node does not conform to Kochut's Constraint (Canonical form) [12], described later, then we do not create this sub-solution in the new population. This step occurs in parallel across our population.

2.3 Fitness Evaluation

Just before each validly constructed sub-solution is added to a new population, we evaluate its hypercube state in order to assign a fitness value. In our implementation, the fitness evaluation of sub-solutions takes place within the construction step. Our construction step is in parallel so fitness calculations are also in parallel, which is important since our fitness calculations are relatively expensive.

2.4 Selection

SIB is a classic combinatorially explosive problem. Constructing all valid sub-solutions as described above for many successive levels in the search tree will predictably result in our process running out of resources; RAM currently being the most important. In order to obtain solutions in a reasonable amount of time, we limit the maximum number of solutions (branches) that are retained from any iteration to a fixed number that stays within available RAM. We use a stochastic selection method to reduce the population of sub-solutions to the beam width. We take a number of random samples of sub-solutions and discard the least fit. We continue to do this while the population is greater than the beam width. Once the population of sub-solutions equals the beam width, we revisit the construction step above.

3 Details

In our process of extending solutions within the structure of beam search, we add nodes at only one end of the snake or coil sub-solutions. We call this end the head. This differs from the literature; however we feel that the growing end is intuitively best described as the head, and the static end as the tail. The static end in our process is always node 0. This is a common approach for reducing symmetry under Kochut's Constraint [12].

In our process, during the construction step, we create a new sub-solution by adding new valid next nodes to the sub-solution. Nodes are valid to be added if they are adjacent to the head and available. Available nodes are nodes that are not in the sub-solutions and have not been disqualified from future addition to the sub-solution by being adjacent to the path at some point other than the head or tail. The valid addition must also pass a check based on Kochut's Constraint before being added. Kochut's Constraint is a symmetric duplication reduction technique. If multiple nodes are valid additions to a sub-solution, then multiple new sub-

solutions are created in a new population, one for every new head node added. If no nodes can be added then, no sub-solution will result in the new population from this parent snake / coil sub-solution. Thus each snake / coil sub-solution can become many, one, or zero new sub-solutions in a new population of one node longer sub-solutions. When no sub-solution can be extended any further, the process terminates. All snakes in this concluding generation are by definition maximal snakes or coils and the longest found in this 'run' of our implementation.

3.1 Kochuts's Constraint

We use Kochut's Constraint as an additional restriction on what constitutes a valid next node to be added to a sub-solution for both snakes and coils. This has been proven previously to correctly eliminate isomorphic duplication. Unsurprisingly, using the constraint in our experiments had an appreciable effect on results. From an implementation standpoint, we simply never create solutions that do not abide by Kochut's Constraint.

Kochut's Constraint as applied in our implementation:

- Start all sub-solutions at the 0 node.
- Create only sub-solutions that conform to the canonical representation. See the UGA AI Website for the definition of canonical form [2].

3.2 Coil

From the standpoint of our process; a coil is exactly like a snake except that it has a known end point. A coil's last node must be adjacent to its origin node. All of our snakes / coils begin at 0, so a coil must end in nodes adjacent to the 0 node. We call these nodes adjacent to the origin of a coil sub-solution, coil nodes [2].

If during the construction step in our process, a valid node to be added to a sub-solution is a coil node, then the resulting sub-solution is not added to the new population. It is a maximal coil

solution because no other nodes can be validly added to this solution after the coil node is added. If a maximal coil is a long enough solution to be of interest, it is reported. This is identical to the coil termination process that Kochut described [12] when he exhaustively searched dimension 7 for snakes and coils.

3.3 Available Nodes as Primary Fitness Component

Many non-exhaustive searches / heuristics have used available nodes as the fitness value or a component of a fitness value / payout [15] [27]. Available nodes are also foundational in our approach as well. The number of available nodes remaining is an excellent way to value the ability of a sub-solution's potential to grow. In fact the available nodes remaining are by definition the upper bound for how much longer a given sub-solution can be extended. Our early approaches to the problem using available nodes as a fitness function combined with a beam width of 100,000 yielded snakes with 94 lengths in dimension 8. This is in contrast to identical trials with random selection yielding snakes with lengths of 77.

3.4 Reachable Available Nodes are Better Still

While the quantity of available nodes is a good component of fitness, it can be improved. The first question we ask is, "Are all available nodes reachable?" The answer is no. Consider Carlson's record length 98 snake in dimension 8 [14]. This solution includes 3 nodes which are available but cannot be reached. We call these isolated nodes. We reduce the available node count for sub-solutions by the isolated nodes. In our experience, as dimensionality increases, isolated nodes are more common and more difficult to avoid. Reachability takes on even more importance as dimensionality of the problem increases. We use the number of reachable available nodes as the primary component of fitness for the lengthening sub-solutions.

Starting with available nodes, we do a depth first search from the head of the snake / coil sub-solutions through the set of available nodes. Only available nodes that can be reached from the head node in a depth first search are reachable available nodes. The number of reachable available nodes is an improvement as a fitness value over available nodes. Testing for reachable available nodes immediately detects nodes that can no longer become part of an eventual maximal snake or coil. This information allows our process to preferentially select for continued exploration other sub-solutions with a similar number of available nodes, but with fewer isolated nodes. With the number of reachable available nodes as the primary fitness component we achieve 97 length snakes in dimension 8 with a beam width of 400,000.

Ascertaining reachability of the available nodes has O (|V|*|E|) time complexity where |E| is the hypercube dimensionality and |V| is the number of available nodes. The number of available nodes is a continually decreasing quantity that we believe should be gathered for calculating even the most basic fitness value. Improving on the available node count by using reachable available nodes has proven to be a differentiator worth the computational expense, which allows us to carry better solutions forward for further exploration.

3.5 Coil Search Optimization

As we have discussed, in order for a sub-solution to become a coil, there must be coil nodes left in the sub-solution's set of available nodes. Since we already calculate the set of nodes that are reachable from the current head node of a coil sub-solution, it is a constant time step to verify that the coil nodes are reachable. If a newly constructed sub-solution contains (Java HashSet ".contains") at least one coil node in the set of reachable available nodes, then at least one coil node remains reachable. The solution is then added to the new population and subjected to selection. If the solution does not contain at least one coil node, it is discarded as it cannot be a

coil. This dramatically reduces the search space and thus the branching factor that our search must cope with. The performance of our coil search predictably improves. This is a natural extension of Kochut's coil search logic when combined with our reachable available node fitness measure.

3.6 Connectivity (Skin) Measure: Contribution to Fitness

We define skin or skin nodes as the nodes that are adjacent to the path, and not adjacent to the head or tail of the snake. Skin nodes are disqualified from becoming part of the growing snake or coil sub-solution. We often think of the SIB problem as a skin sharing problem. That is, maximizing the number of times a disqualified node is used as skin by snakes and coils.

Once we have the set of reachable available nodes for a sub-solution, we perform one final assessment. As a selection tie breaker, we poll the reachable available nodes to determine how many other available nodes each reachable available node is adjacent to. From this we determine the number of unavailable nodes (skin nodes) that each must be adjacent to. For example in dimension 8, each node has 8 adjacent nodes. If 2 of these are available, then the skin count would be 6. We take this count of skin for every reachable available node in a sub-solution and sum it.

High numbers of skin connections among a sub-solution's reachable available nodes suggest a low cost in reachable available nodes in future extensions. We use this value as a tie breaker between solutions with equivalent reachable available nodes. If two solutions have equivalent numbers of available nodes, our process preferentially selects for future exploration those solutions with available nodes more connected to skin nodes.

3.7 Stochastic Selection

When the number of candidate solutions has grown larger than the beam width through the construction step discussed above, we reduce the number of solutions to the beam width. Through an iterative stochastic method we randomly select a number (tournament size) of sub-solutions and discard the least fit individual. This operation is similar to GA tournament selection, but in reverse. We call this process reverse tournament selection.

We experimented with a strictly greedy approach to selection before arriving at the GA inspired stochastic reverse tournament selection process. Results improved for selection with some randomness. For example, in dimension 8, with a strictly greedy approach of retaining the highest ranking finesses within the beam, and a beam size of 100,000, we achieved snakes of length 96. When the stochastic element is added to selection we achieved snakes of length 97. Similar improvements were seen in all trials of dimensions 6-13.

In our approach to calculating fitness, the very best solutions at times in their development may not have the highest fitness. If this were not the case, then a strictly greedy approach would yield the best results. Of note, Beam Search with stochastic selection repeatedly delivers solutions of the same length, given a specific beam width, while the actual snakes / coils that are found may vary each time the program is run.

3.8 Hardware

We ran trials and set records on several different multicore machines. A circa 2005 server class machine with 64 GB RAM and 16 cores, a Virtual Machine running on modern equipment with 26 much faster cores (2100 MHz), and 64 GB RAM, and a personal laptop with 8 cores and 22 GB ram. While records were initially broken with other machines, all of the new SIB lower

bounds presented here were set on the Virtual Machine with 26 dedicated cores, hosted on a modern server class machine provided by the University of Georgia Computer Science IT Group.

Beam search is readily parallelizable, and we took advantage of this. None of the records set, or trials run took more than 5 days with 26 cores. For example the new coil lower bound set in dimension 12 took 2 days of actual time and over 1300 hours of core time (2.3 days * 24 Hours * 26 Cores* 95% avg. core usage). This record setting run utilized at most 59 GB of RAM with a beam width of 50,000 sub-solutions.

Table 2: Snake and Coil Records for Non-Exhaustively Searched Dimensions 8 – 13

Dimension	Length	
	Snakes	***Coils***
8	98	96
9	190	188
10	370	(348) / 358
11	(695) / 705	(640) / 666
12	1280	(1238) / 1268
13	2466	2468

3.9 Representation

We represented the sub-solutions as a linked list of integers matching the nodes in the path. This approach was very memory intensive and we used as much as 64 GB in our trials. Memory is the current limiting factor of our process and implementing a more efficient representation such as the bit map approach described by Diaz-Gomez [28] is a high priority.

4 Results

We found new lower bounds for snakes in dimension 11 and coils in dimensions 10, 11, and 12. We matched the lower bounds for snakes in dimensions 8, 9, 10, 12 and coils in 8 and 9. In Table II., from [23], the numbers in parenthesis are the previous lower bounds before our work.

An up to date list of snake, and coil absolute and lower bounds is maintained at The University of Georgia SIB Records Page [23]. An example solution for each new lower bound is included below.

Examples of Record Snakes and Coils:

Snake, Dimension 11, length 705

01234523103253123452310326425430134523402452103476430125135430134523410640325312402345246297230432543103453152103456123451347145234024516015413423013504301356273241352432017430A913218A152032541053612704325410423125162042345312345042543217043056125420342135427

Coil, Dimension10, Length 358

01203143054314520362054206314360530410213053412073063253054235014305325430526501203143054327823450341302105625034523503410532450352362764351430543204360320435032041702765235406502391345034130210562503452350341053245035236037024135031201403506341360245026302541345034130213873120314305431425036025420631436035041021305314574302150310632452153256305126324135218

Coil, Dimension 11, Length 666

012031430543145203620542063143605304102130531420724134503413542632413503452431245210341087014301254213425430531423624502105342530543127142543053145305263241345034134523143653413923143054314270241350312014035063413602450263025413450341307803143054314520362054206314360530410213053475035431201354634102130531423624502462542143641A5041021326120314305431452036205420631436243147041345034135895314305431427024135031201403506341360245026302541345034130780314305431452036205420631436053041021305347530542031024523402563453052895103413623143054314520780361034102871452430631781360342041841A1405426A24502943594054159461541394A3145120523A93253975362786204814054953405427843041

Coil, Dimension 12, Length 1268

012031430523041342305206230253452145041301204702503413025034603510341021450413012362145218241254352032602503243140325034130216521031437103412435241250352142634502103140541254081402143015341301295210314054125430153624125305214253421438234521450260421403105271534521450413012046025034130250342041254824125435264250310214036901526153BA351625741031423764021031405412014624130250318640325034130250423052062302534521450371304124350324125305216510314052862412543520326541021430153413021952103140541254301536241253052142534214382345214502604214031052715345214504130120460250341302503420412548241254352642503102140369052A76421450413012045362412530521403693052A7642145041301204536241253052

142534572450264021674039630412046205412543528952345214502604214031052715241034
054125049204120642054125432092405214523503B4052103BA3012508798052013AB310250
3674BA4723102503AB3052019B32630520159205264206103546350234051 2B0367BA723A92
387206703783971684502578520560120327824 37B362BA260A241206435145205B12052643B
A439A65A93A1631024920124578A1

5 Conclusions

Our approach was largely inspired by efforts in [14], [17], [28], [18], and [11] to use a GA to improve known lower bounds on this difficult problem. The GA will usually yield good solutions to the SIB problems, and by applying more computation effort, the solutions can be improved to a point, however; the GA applied to the SIB problem runs into diminishing returns.

Our approach to fitness relentlessly selects dense risky solutions that are highly likely to dead end, but are the only prospects for an excellent result. We generate tens of thousands of these 'risky' solutions expecting the vast majority of them to fail. The few that don't fail are expanded.

Symmetrical redundancy could have degraded our process by clogging it with functionally identical solutions. Fortunately we were able to avoid considering any solution that does not adhere to Kochut's Constraints; In effect making this portion of the search space 'invalid'.

The Beam Search is a very simple technique to implement. There are few parameters to set. No extensive tuning is required. The only parameters settable in our implementation were the reverse tournament size and beam width. The beam width, while settable, is bounded by memory constraints and, predictably, larger beam widths produce better results. Additionally, our process requires no backtracking or recursive calls.

Determining which available nodes are actually reachable further improves our ability to predict a sub-ultimate quality. Formerly undetectable 'exclusive or' forks in the search space are now obvious as soon as one of the exclusive paths is taken, allowing us to preferentially choose sub-solutions without such a flaw. Reachability as it applies to coil nodes further shrinks the search space by permitting us to discard potential coil sub-solutions without loss when its coil nodes become unreachable.

6 Future Work

6.1 Representation

Our current implementation is limited because of our list representation. We plan to move to a Diaz-Gomez bit-map type approach [28]. This will greatly reduce the memory requirements for a given beam width. We anticipate being able to increase the beam width by at least a factor of 100.

6.2 Concurrent Stochastic Selection

Selection is not concurrent in our implementation. Once we improve the representation allowing us a dramatic increase in beam width, an increased and the number of solutions will be pruned in each iteration. This will make a lack of concurrency in selection more of an issue. We plan to implement a method that performs stochastic selection in parallel; possibly within buckets, or partitions of a randomized population list.

6.3 Fitness Function

In our future work we plan to explore more graph theory based improvements to our fitness function. We have some ideas for detecting unusable dead ends that can be reached but not returned from and how detecting these cases may help improve the predictive value of our fitness function.

ACKNOWLEDGMENT

We would like to thank Dustin Cline and Thomas Horton for their important contributions to this project, Snake-in-the-Box research, and the Institute for Artificial Intelligence. This study was partially supported by resources and technical expertise from the University of Georgia Computer Science Department.

REFERENCES

[1] M. Garey and D. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, San Francisco: Freeman, 1979.

[2] W. Potter, "Snake-In-The-Box Dictionary," 2014. [Online]. Available: http://ai1.ai.uga.edu/sib/sibwiki/doku.php/dictionary. [Accessed 08 September 2014].

[3] W. Kautz, "Unit-distance Error-Checking Codes," IRE Trans Electronic Computers, Vols. EC-7, no. 2, pp. 179–180, 1958.

[4] V. Klee, "What is the Maximum Length of a d-Dimensional Snake?," The American Mathematical Monthly, vol. 77, no. 1, pp. 63-65, 1970.

[5] A. Hiltgen and K. Paterson, "Single Track Circuit Codes," IEEE Transactions on Information Theory, vol. 47, pp. 2587–2595, 2000.

[6] F. Zhang and H. Zhu, "Determination of optimal period of absolute encoders with single track cyclic gray code," in Journal of Central South University of Technology, New York, 2008.

[7] M. Blaum and T. Etzion, "Use of snake-in-the-box codes for reliable identification of tracks in servo fields of a disk drive". U.S. Patent 6496312, 2002.

[8] Y. Yehezkeally and M. Schwartz, "Snake-in-the-Box Codes for Rank Modulation," in Proceedings of the 2012 IEEE International Symposium on Information Theory, July 1-6, Cambridge, Piscataway, 2012.

[9] I. Zinovik, Y. Chebiryak and D. Kroening, "Periodic Orbits and Equilibria in Glass Models for Gene Regulatory Networks," IEEE Transactions on Information Theory, vol. 56, no. 2, pp. 805–82, 2010.

[10] D. Davies, "Longest 'separated' paths and loops in an n cube," IEEE Transactions on Electronic Computers, vol. 14, p. 261, 1965.

[11] W. Potter, J. Robinson, J. Miller and K. Kochut, "Using the Genetic Algorithm to Find Snake- In-The-Box Codes," in In Proceeding of the 7th International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems, 1994, May 31 - June 03, 1994, Austin, Newark, 1994.

[12] K. Kochut, "Snake-In-The-Box Codes for Dimension 7," Journal of Combinatorial Mathematics and Combinatorial Computing, vol. 20, pp. 175–185, 1996.

[13] D. Kinny, "New Approach to the Snake-In-The-Box Problem," in ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012, Amsterdam, 2012.

[14] B. Carlson and D. Hougen, "Phenotype feedback genetic algorithm operators for heuristic encoding of snakes within hypercubes," in Genetic and Evolutionary Computation Conference, GECCO 2010, Proceedings, July 7-11, Portland, New York, 2010.

[15] D. Kinny, "Monte-Carlo Search for Snakes and Coils," in 6th International Workshop, MIWAI 2012, Ho Chin Minh City, December 26-28, Proceedings, New York, 2012.

[16] P. Brooks, "Particle Swarm Optimization and Priority Representation (MS Thesis).," Artificial Intelligence, University of Georgia at Athens (201012), 2012.

[17] D. Casella and W. Potter, "Using Evolutionary Techniques to Hunt for Snakes and Coils," in Proceedings of the IEEE Congress on Evolutionary Computation, September 2-4, 2005, Edinburgh, UK, Piscataway, 2005.

[18] M. Juric, W. Potter and M. Plaskin, "Using PVM for Hunting Snake-In-The-Box Codes," in Proceedings of the 7th Conference of the North American Transputer Users Group, October 23-25, Athens, Amsterdam, 1994.

[19] J. Bishop, "Investigating the snake-in-the-box problem with neuroevolution," Dept. of Computer Science, The University of Texas at Austin, 2006.

[20] L. Adelson, R. Alter and T. Curtz, "Long Snakes and a Characterization of Maximal Snakes on the d-cube," in Proceedings of the Fourth Southeastern Conference on Combinatorics, Graph Theory, and Computing, Florida Atlantic University, Boca Raton, March 5-8, Winnipeg, 1973.

[21] H. Abbott and M. Katchalski, "On The Construction of Snake In The Box Codes," Utilitas Mathematica, pp. 97-116, 1991.

[22] E. Wynn, "Constructing circuit codes by permuting initial sequences," 8 January 2012. [Online]. Available: http://arxiv.org/abs/1201.1647. [Accessed 5 June 2014].

[23] W. Potter, "SIB Records," 2014. [Online]. Available: http://ai.uga.edu/sib/sibwiki/doku.php/records. [Accessed 5 June 2014].

[24] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, New York: Prentice-Hall, 1995.

[25] D. Howe, "Beam Search," 3 December 2007 . [Online]. Available: http://foldoc.org/beam+search. [Accessed 1 June 2014].

[26] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach: 3rd Edition, Upper Saddle River: Prentice Hall, 2010.

[27] D. Tuohy, W. Potter and D. Casella, "Searching for Snake-in-the-Box Codes with Evolved Pruning Models," in Proceedings of the 2007 International Conference on Genetic and Evolutionary Methods, GEM 2007, June 25-28, Las Vegas, Athens, 2007.

[28] P. Diaz-Gomez and D. Hougen, "Genetic Algorithms for Hunting Snakes in Hypercubes: Fitness Function Analysis and Open Questions," in Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2006), June 19-20, Las Vegas, Los Alamitos, 2006.

CHAPTER 3

FINDING LONGEST PATHS IN HYPERCUBES: 11 NEW LOWER BOUNDS FOR SNAKES, COILS, AND SYMMETRICAL COILS[2]

[2] Meyerson, S., Drapela, T., Whiteside, W., and Potter, W.D., "Finding Longest Paths in Hypercubes, 11 New Lower Bounds: Snake, Coils, and Symmetric Coils," Proceedings of the 28th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, (IEA/AIE'15), Springer Verlag, Seoul, Korea (To Appear) 2015 – Reprinted here with permission of publisher.

Abstract

Since the problem's formulation by Kautz in 1958 as an error detection tool, diverse applications for long snakes and coils have been found. These include coding theory, electrical engineering, and genetics. Over the years, the problem has been explored by many researchers in different fields using varied approaches, and has taken on additional meaning. The problem has become a benchmark for evaluating search techniques in combinatorially expansive search spaces (NP-complete Optimizations).

We build on our previous work and present improved heuristics for Stochastic Beam Search sub-solution selection in searching for longest induced paths: open (snakes), closed (coils), and symmetric closed (symmetric coils); in n-dimensional hypercube graphs. Stochastic Beam Search, a non-deterministic variant of Beam Search, provides the overall structure for our search. We present eleven new lower bounds for the Snake-in-the-Box problem for snakes in dimensions 11, 12, and 13; coils in dimensions 10, 11, and 12; and symmetric coils in dimensions 9, 10, 11, 12, and 13. The best known solutions of the unsolved dimensions of this problem have improved over the years and we are proud to make a contribution to this problem as well as the continued progress in combinatorial search techniques.

Keywords: stochastic beam search, snake-in-the-box, combinatorial optimization, graph search, hypercube, heuristic search

1 Introduction

The Snake-in-the-Box problem (SIB) is a computationally hard problem concerned with finding the longest induced path through an n-dimensional hypercube (or n-cube) graph. Finding longest induced paths is well known to be NP-complete. A hypercube graph consists of a set of binary numbered nodes $\{0 \ldots 2^{n-1}\}$ in which two nodes are adjacent if and only if their binary values differ by exactly one bit. An induced path (achordal path) is a path in which each node in the path is adjacent only to its immediate predecessor and successor nodes within the path. In the SIB problem, open achordal paths are called snakes, and closed achordal paths are called coils. For snakes, any path which cannot be extended is called a maximal path, with the *longest* of these being the longest maximal path. The longest maximal snake for any n-cube is the snake solution (absolute bound) for SIB in dimension-n. Similarly, the longest possible coil for any n-cube is the coil solution (absolute bound) for dimension-n. Symmetric coils are a special case of coil, where the transition sequence for the first half of the path is equal to the transition sequence for the second half of the path. Transition sequence notation describes paths using the bit positions that change between two nodes. Figure 1 shows the symmetrical coil described by the node sequence {0, 1, 3, 7, 6, 4, 0} and the transition sequence 012012. For these and other SIB terms and definitions, please visit the UGA Institute for Artificial Intelligence's webpage [1].

Introduced by Kautz in 1958, the Snake-in-the-Box problem has been extensively studied for over 50 years. The problem has found relevance in many fields including coding theory [2], electrical engineering [3], analog to digital conversion [4], precision high speed rotational measurement devices [5], disk sector encoding [6], rank modulation schemes for charge

distribution in multi-level flash memories [7], and genetics related to embryonic development [8].

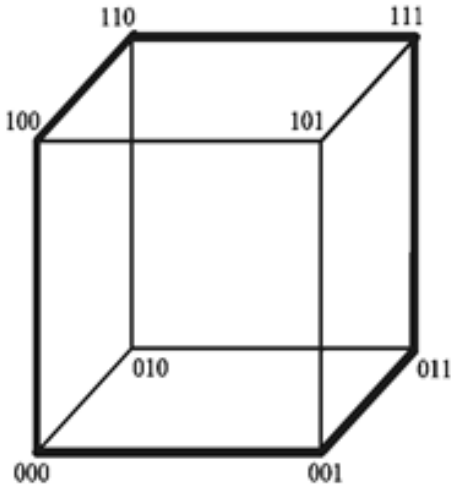

Figure 1: 3-dimensional hypercube (with a symmetrical coil in bold)

For dimensions n < 7, snake and coil solutions were computed via exhaustive search by Davies [10]. Solutions for n = 7 were determined via genetic algorithm by Potter [11] and later verified by the exhaustive techniques of Kochut [12] which we will be building upon. See table 1, below.

For dimensions n > 7, exhaustive search remains infeasible for finding solutions due to the exponential growth of the problem as dimensionality increases. It is estimated that it would take many years of computing time to exhaustively search dimension 8 [12]. In lieu of this, non-exhaustive techniques, both computational search and mathematical construction based approaches have been applied to improving the known lower bounds of higher dimensions. These include: genetic algorithms [10], hybrid genetic algorithms [13], nested Monte Carlo Search schemes [14], particle swarm optimization [16], population-based stochastic hill-

climbers [16], distributed heuristic computing [17], neural networks [18], hybrid evolutionary algorithms [16] and mathematical constructive techniques [19, 20, 21].

Table 1: Current SIB Absolute Bounds

Dimension	Length		
	Snakes	*Coils*	*Symmetric Coils*
1	1	0	0
2	2	4	4
3	4	6	6
4	7	8	8
5	13	14	14
6	26	26	26
7	50	48	46

With each successive technical improvement, enhanced by increases in computing power, the lower bounds for these higher dimensions continue to be improved. For solution lengths of n-snakes, n-coils, and n-symmetrical coils for $n \leq 7$, as well as current lower bounds non-exhaustively searched for $8 \geq n \leq 13$ see [22].

2 Overview of Our Previous Work

Building on our previous work [23], we again implement our non-deterministic variant of Beam Search, which we refer to as Stochastic Beam Search, which combines elements of graph search with methods common to evolutionary computing. Our Stochastic Beam Search (SBS) differs from BS solely in how the decision to prune is made. Whereas BS retains only the absolute best sub-solutions at each step, SBS provides for randomness similar to the tournament selection process commonly used in Genetic Algorithms (GA) to determine which sub-solutions

are retained. Basic tournament selection retains only the single best sub-solution from among a randomly selected subset of a population. SBS tournament selection, on the other hand, is a reverse tournament selection that discards only the single worst sub-solution found in each tournament. Tournaments continue to be run while the population of sub-solutions exceeds the beam width. Figure 2 shows a beam search with a beam width of 2 progressing through successive expansions of selected sub-solutions. Each column shows the number of candidate sub-solutions generated in that iteration, with the circled sub-solutions representing the candidate sub-solutions that have been selected for continued exploration in the next iteration. The unselected (un-circled) solutions are discarded. Notice there is no selection while the number of leaf nodes is less than the beam width.

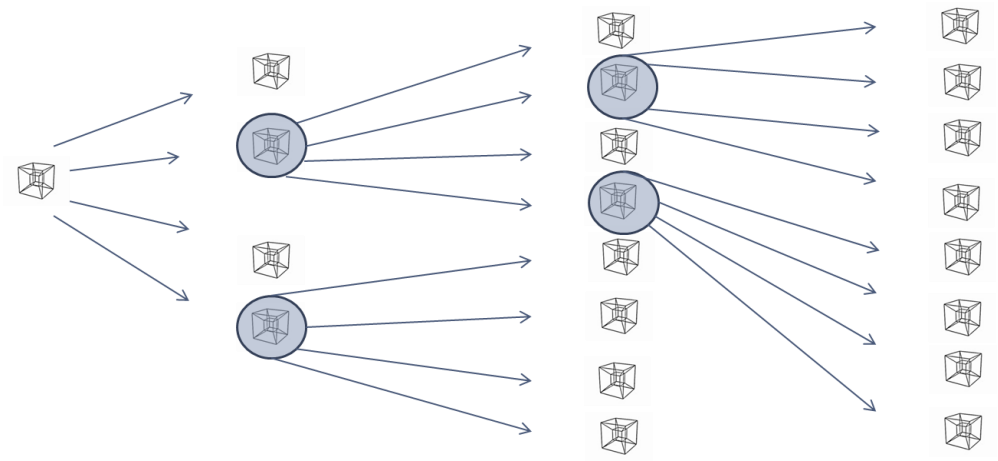

Figure 2: Beam Search

In our previous work, we extended the excellent "available nodes" metric [14] [25] by determining which available nodes were actually reachable from a sub-solution's head. We called these "reachable available nodes". Our base implementation in our previous work searched only for snakes. To search for coils we added an additional check to our base fitness

function which allowed us to discard any sub-solution that did not retain the possibility of becoming a coil based on end point existence. For more information on our previous path exploration optimization and fitness function see [23].

3 Improvements to Representation and Selection

The beam width is constrained by RAM for practical reasons. In order to use a wider beam that considers many more sub-solutions, we needed to reduce our application's memory footprint. We improved our sub-solution representation from a linked list to a bitmap in which a sub-solution is stored as an array of bits rather than a list of integers or shorts. This new representation offered a huge improvement over the previous linked list. Using this representation netted a memory savings of greater than 96%. A representation like this was first suggested by [26]. This improvement allowed us to dramatically increase our beam size, and moved the bottleneck of our process from memory to processing.

Because of this improvement in our representation we were able to increase the beam width from 50,000 to 1,000,000 sub-solutions for searches in dimension 12. Correspondingly, we improved on our previously reported new lower bound in dimension 12 [23] by 8 nodes.

4 Improvements to Fitness for Coil Search

Here we present our latest fitness improvements for searching for coils. Our symmetrical coil search is addressed in Section 5.

4.1 Removing Dead End Nodes

In our previous work, we introduced connectivity values for reachable available nodes [23]. The connectivity value of a reachable available node is equal to the number of reachable available nodes that are adjacent to it. *Dead End Nodes* are reachable available nodes with only one reachable available node neighbor, and thus have a connectivity value of 1. Dead end nodes may only become: (1) the terminating node of a snake, (2) a skin node to its sole reachable available node neighbor, or (3) an isolated node [23].See figure 3 below.

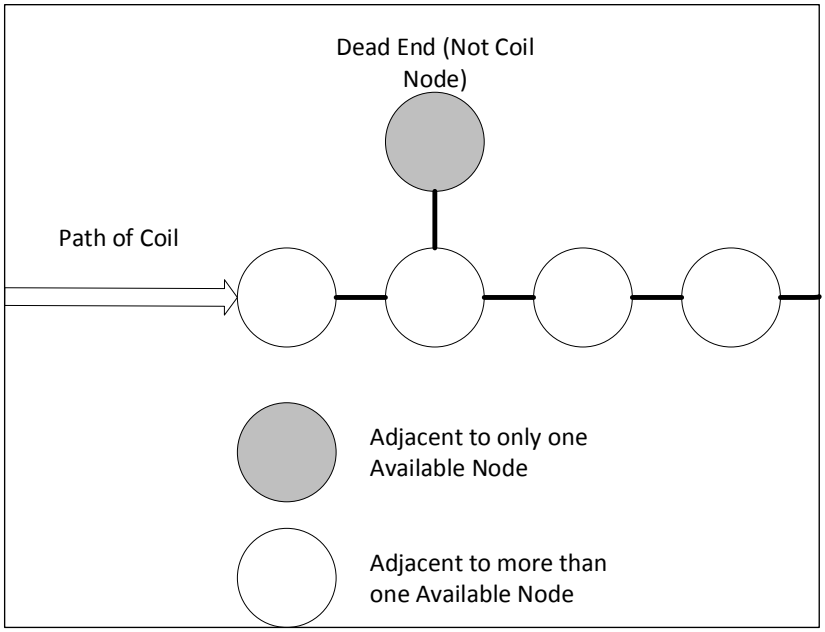

Figure 3: Dead end nodes.

Dead ends correlate to exclusive OR forks in the pool of reachable available nodes. These are points where large numbers of nodes that were previously additive to the fitness value as reachable nodes, become isolated once the path of the snake or coil passes this fork. Being able to efficiently detect and measure the impact of an exclusive OR fork as soon as it appears in a

sub-solution's hypercube is of great predictive value in determining which sub-solutions should be carried forward for further consideration (fitness and selection).

In coil search, the end points of our search process are adjacent to our start node (node 0). We call these nodes coil-forming nodes [23]. For coils, any dead end node that is not a coil-forming node is an unusable dead end and cannot be part of a valid coil solution. We remove these nodes from the total of reachable available nodes that serves as our primary fitness value. Removing dead end nodes from reachable available nodes in our fitness calculation, with a beam width of 100,000, yielded length 92 coils in dimension 8.

4.2 Extension of Dead End Concept - Blind Alleys

A natural next step to identifying and removing dead end nodes is to look for sub-paths which lead without alternatives to dead ends. We call such paths blind alleys. See Figure. 4. Any reachable available node that is adjacent to a dead end node and only one other reachable available node is a blind alley node. Furthermore, any node that would become a dead end upon the removal of a dead end node is also a blind alley node. In coil search, we have known end points, so we can remove all dead ends that are not coil nodes without concern.

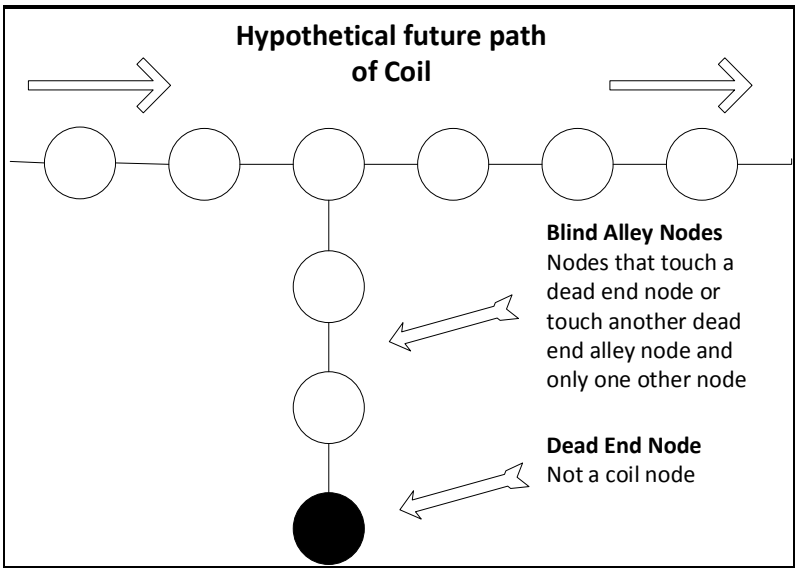

Figure 4: Blind Alley Nodes

Reducing our fitness value by the number of blind alley nodes can be done efficiently. For each dead end node removed, we need only determine if its reachable available node neighbor(s) has become a dead end node. If so, we remove this node from the sub-solution reachable available node set. We repeat this process until no more dead end nodes are found in the sub-solution graph.

Once we have completed the removal of unreachable available, dead end, and blind alley nodes, the remaining nodes form a much better basis for fitness than available nodes alone. When we use a fitness value refined by removing dead end and blind alley nodes from the count of the reachable available nodes set, we are able to routinely match the record of length 96 coils in dimension 8 with a beam width of only 100,000 sub solutions.

4.3 Concurrent Stochastic Selection

Because our improved representation enables us to increase the beam width of our search, our previous serial selection process becomes a pressing core utilization bottleneck. To address this constraint, we implement a concurrent selection process. We first randomize the order of our population of sub-solutions, then partition the population and conduct separate concurrent tournaments within each partition; reducing the population of each partition until the total population of all partitions is less than or equal to the beam width. We first randomize the order of the population of sub-solutions because, without this step, the more closely related sub-solutions are to each other, the closer in the population list they appear. This would create a positional bias that has nothing to do with our mission of finding the best possible solution. We found performance to be better when we incurred the small computational expense of randomizing the order of the population list before partitioning the list and performing selection.

4.4 Hardware

We ran trials using several different multi-core systems: a virtual machine backed by 48 cores and 100 GB RAM; multiple nodes of a GACRC cluster with 12 and 32-core processors with up to 64 GB of memory; and a personal laptop with 8 cores and 22 GB RAM.

Our current implementation utilizes processing resources at very close to 100%. The only serial parts of our process are the randomization, partitioning, and recombination steps of the selection process; and the partitioning and recombination steps of the sub-solution construction process.

Our most intensive search (processors × time) was for a new coil lower bound for dimension 13. This was achieved on a GACRC 32-core computing node with 64 GB RAM available and completed in 28 days, or 870 core days; calculated as core-count × core-usage% × time with an estimated core-usage of 97%. Less than half of the available RAM was used.

5 Specific to Symmetric Coil Search

Our long symmetric coil finding implementation is based on our long coil finding approach, and differs from previous approaches that either grow individual n-dimensional snakes and then attempt to "double" them to form coils / symmetrical coils in n+1 [21], or search for snakes and occasionally find coils.

In our process we specifically set out to find / build long symmetrical coils. Within a single sub-solution, we simultaneously "grow" two valid, bit transition identical, sub-snakes in a single hypercube. Within this sub-solution, we refer to these sub-snakes as snake1 and snake2. When we refer to the sub-solution, we are referring to both snake1 and snake2 together. The objective is to link the head of snake1 to the tail of snake2, and the head of snake2 to the tail of snake1 in order to create a long valid symmetrical coil.

To remain valid, neither snake1 nor snake2 may violate the others or its own achordal path constraints. In addition sub-solutions that are retained after each construction step retain the potential to be linked; one or more of snake1's available coil-forming nodes remains reachable by snake2's head node, and one or more of snake2's available coil-forming nodes remains reachable by snake1's head node. There is one last constraint we impose on the growing sub-

solution. Snake1 must conform to our previous implementation of Kochut's symmetry reducing constraint [11] [23].

During each construction step, new valid sub-solutions are constructed from a previous sub-solution by adding a single node to the head of snake1. Then a new valid node is added to the head of snake2. This new addition to snake2 must share the bit shift position of the node added to snake1 (relative to the previous node in snake1). We call this relationship of the addition of a new head to snake1 and the next addition to snake2 a symmetrical move. If no valid symmetric move is available for snake2 then this partially constructed and invalid child is discarded.

In contrast to our coil and snake process, we initialize our process with more than a single sub-solution. We create an initial population containing every valid sub-solution in which snake1 starts at node 0 and snake2 starts at some other valid node in the hypercube. Sub-solutions are not created where snake1 and snake2 are adjacent. If the two nodes are adjacent, then this would represent an achordal constraint violation.

5.1 Construction Phase

For every sub-solution in the current population, and for each available node that can be validly added to snake1 as a new head, we perform the steps below.

1. For each reachable available node adjacent to the current head of snake1: If snake2 has a valid symmetrical move, we add the corresponding node to be the new head of snake2; otherwise, the new sub-solution is discarded, as it can never form a symmetrical coil.
2. If the node newly added to snake1 is one of snake2's coil-forming nodes, the sub-solution has terminated in a symmetrical coil solution. Symmetrical coil solutions long enough to be of

interest are reported. If the sub-solution (now two nodes longer than its parent) has not terminated, it is added to the successor population.

3. After all members of the current population have been processed to this point, the successor population replaces the current population.

5.2 Fitness evaluation phase

For every sub-solution in the successor population, we perform the following steps in order to compute its fitness value:

1. Find the set of available nodes reachable from the head of snake1, and separately the set of available nodes reachable from the head of snake2. As we do for regular coils, we allow coil-forming nodes to be added to the reachable available node sets; however, reachability exploration is not continued from these nodes. As a coil-forming node is the last node to be added to any coil solution, any nodes reachable only through adjacency to it are effectively unreachable in the context of coils.
2. If none of snake2's coil-forming nodes are in the reachable available node set of snake1, we discard the sub-solution; as it can never form a symmetrical coil. This one constant time check drives a large reduction in the number of solutions considered.
3. Ascertain if the two sets of reachable available nodes are equal or disjoint. Because of the symmetry of the hypercube and of our search, the reachable available node sets of snake1 and snake2 will either be equal or disjoint. If equal, all reachable available nodes are reachable from the heads of snake1 and snake2, and our fitness evaluation will not require adjustment. If disjoint, the set of all reachable available nodes is split into two distinct subsets, which (again because of symmetry) are functionally identical. To account for this,

we will double the fitness value at the end of the fitness evaluation phase. In either case (disjoint or equal), because of symmetry, we only need to work with Snake1's reachable available node set from this point forward.

4. Remove blind alley nodes from snake1's reachable available node set using snake2's coil-forming nodes as our end points; just as we do for standard coils.
5. Calculate a connectivity value as in [23] from the remaining nodes in snake1's reachable available node set.
6. Lastly, double the fitness value if the reachable available node sets were determined to be disjoint in step 3.

5.3 Termination

As previously described in all of our approaches, the construction and fitness evaluation processes continue until there are no longer any sub-solutions in the population.

6 Results

We found new lower bounds for snakes in dimensions 11, 12, and 13; coils in dimensions 10, 11, and 12; and symmetrical coils in dimensions 9, 10, 11, 12, and 13. Table 1 shows current lower bounds for dimensions 8–13. Parenthetical values indicate previously known lower bounds. Asterisks indicate where we have improved on our previously published [23] records. An up to date list of snake and coil, absolute and lower bounds, is maintained at The University of Georgia SIB Records Page [22]. Please find examples for lower bounds from our results for snakes, coils, and symmetrical coils in section 9. Table 2 contains lower bounds for non-

exhaustively searched dimensions of SIB. The numbers in parenthesis indicate the previous record before our work. The numbers with asterisks indicate where we have surpassed our previous record lower bounds.

Table 2: Current SIB Lower Bounds

EXHAUSTIVELY SEARCHED DIMENSIONS 8 - 13.

Dimension	Length		
	Snakes	***Coils***	***Symmetric Coils***
8	98	96	94
9	190	188	(180) 186
10	370	(*358) 362	(340) 362
11	(*705) 707	(*666) 668	(640) 662
12	(1280) 1302	(*1268) 1276	(1128) 1222
13	(2466) 2520	2468	(1898) 2354

7 Conclusions

Our approach was largely inspired by efforts in [13], [16], [25], [17], and [10] to use a GA to improve known lower bounds on this difficult problem. The GA will usually yield good solutions to the SIB problems, and by applying more computation effort, the solutions can be improved to a point; however, the GA applied to the SIB problem runs into diminishing returns. Our use of fitness relentlessly selects dense solutions that are highly likely to dead end, but are the best prospects for an excellent result. We expand upon the small fraction we are able to practically process that we judge through fitness as good prospects. Improvements in the predictive value of our fitness function resulted in new lower bounds in several dimensions and

solution classes (snakes / coils / symmetrical coils) while increasing the beam width process by 50–100 times yielded only small improvements in results.

Exploiting the inherent constraints of specific solution classes (specifically coils and symmetrical coils) in the Snake-in-the-Box problem enabled us to dramatically reduce the effective search space, and to impose powerful checks on the validity of sub-solutions. For example, the symmetrical coil aspect of the SIB problem is the most constraining. We checked that sub-solutions could reach coil-forming end points and thus retain coil forming potential. We also checked sub-solutions conformance to the symmetrical property of symmetric coils. If the sub-solution did not pass these checks, it was discarded. This left us with a small fraction of the search space that we would have had to traverse without the intelligent application of known constraints. This approach explains the particular strengths of our process when applied to coils and symmetrical coils. We are particularly pleased with the improvements achieved in symmetrical coil lower bounds in dimensions 9–13, which saw lower bounds improvements of 6, 12, 22, 94, and 456 nodes respectively.

Beam search is an excellent platform to experiment with approaches to assess the state of the hypercube (fitness functions). Once the beam width is sufficiently large, small changes in the efficacy of the fitness function led to large performance changes. We expect that more algorithms common to graph search may merit application to this problem.

8 Future Work

In future work we plan to explore more graph theory based improvements to our fitness function. One of the most promising we are testing is articulation point / bi-connected

component detection algorithms. The techniques we have described including potential future work with articulation points may lend themselves to searching for snakes using defined end points; and could include techniques like our symmetric coil search, where multiple sub-solutions with many different end points are considered within the same search; much like considering multiple start points for complimentary symmetrical snakes.

9 Examples of Record Snakes, Coils, and Symmetrical Coils

9.1 Snakes

Snake, Dimension 11, Length 707

01234523103253123452310326425430134523402452103476430125135430134523410640325312540234523102532123452310326425430134523402452103476430125135430134523410640325312402345240231861325432135230132543213564254014523402452104710234524023451325230461043254310345315210345298254301251354301345234016403253154320425432017213452340216013254321352301325432135602842304601432543103453152103467430125420432543103452462301325432135230132543210980123452310325312345231032642543013452340245210347643012513543013452341064032531240234524629723043254310345315210345273145106514204325413270123452310325312345231037896012543215203125435270432541042312516402432531234504254321374034231452302514A812319A4631034053103

Snake, Dimension 12, Length 1302

012031401203521403120140365304102130410531041321750120314012035210432631023140135014031201403873041021304105310413201362345304130210413053725014031203523

1402604102130412530210413029720314012035214031201403653041021304105310413207802314013501403120140356304102130413820653052104325305612045012841231401237130410213041361035012514012531263102314012 6A2350140312014035630410213041253021041302792031401203521403120140365304102130410531041320725031401236523104132103653046341021348312035214031201403653041021304105310413217310213401253021041302972031401203521403120140720413085031401235630623106304132167123 14BA01350140312014035630410213041253021041302792031401203521403120140365304102130410531041320725031401236523104132103653046341021348312035214031201403653041021304105310413217310213401253021041302972031401203521403120140720413081401326021041321651231401235869321041 32A8231401236532104130580314012 06B02104130297203140120BA02104132A8234023563204305803402396321041308230140356304103820314012038642314013651B0123AB2A7314750358120143891341047904109302310B256230410BA014031A53104128B21403295230425B2403129A2130426524031AB130410B501402495024032103948214029578639230410 5A0140328A2340246842043210691432403 19B124B3417924B2918BA3823A014534102952049741B70B261201A693123410213290A5380950324139128B64B816B83719012031

Snake, Dimension 13, Length 2520

252001203140120352140312014036530410213041053104132072503140120314035630123140132513041021304138731403120140315231041321036530415210415031401257210413212 6304102130412542046014031201409704102130410531041321046304102130413853140312014035630410213041387314031201403152402103623041251401203140125721041325302130410215630410213041251401203140185104132146540154651273041302104130A8210413021041521403120140365304102130410531041321982312401350140312014035630410213041385314031201403154136531203140350 1

25305721041325302130410215630410213041251401203140387304130210413750210413021046540120314012982104130210456401203140129821041302104564012031401205402130410213803401203473140312035231401AB1041325310230410B4031402804136231401 8B104130210413BA314012031403A9304130210417814012031402704102130410531042314018031403210352314037830413024305402130410840314264102130458403120149541021304102765240314 0C5041304A01403124B21041302104152140312014036530410213041053104132198123140135014031201403563041021304138531403120140315413653120314035012530572104132530213041021563041021304125140120314038730413021041375021041302104654012031401298210413021045640120314012054021304102138204130210416714012031403563041325302130410 2A42031402804102130413743021041691401203140271041021304105310413201463041021304125140120314 03BA3041302104152140312014036410231401350140312014072041302104187140120314035630413253021304137430480140312045021041302104AB401203140120540213041028420347314031203523140365304130210413098342031403BA30413021041A914012031401264014764102130413643021041521403120 1AC1021304854031428C42130418A1403147104130214C15631784234894312019610213065032CB230456403120168102130A503120145941021340854031201465410213 0B80312037204792B72402730213458431420 41BA146A96402134024A046A41203140198104130210590120314026 9C364120318913219501203410 2C631A681541023185132C54620A24023493CA206CA13219A09C4321945A20A358138B40346B432014B65A8541245840341059234124690431691489C3851043204503624069

032019810218C2403146B413821890126421983401298139418041804C70AC62430246A98B630986
402189120149088094102134085403120149A0142130410A241059123195146598694 31A0320
891041A219A0432014A741261471A025A124035A3028A405436B8013502405145642CB6821
804235125930960283608914694C309714703901AB64163A93BA76B230

9.2 Coils

Coil, Dimension10, Length 362

(See Symmetric Coil 10)

Coil, Dimension11, Length 668

012302401230150210340230650320430123051021407540123056032102503201340230150 2
106501235432078530120356032102503210340230150210650123074032105601205103210451
650230560315321098012352054012301502106501230170210420321056012051032043012305
201230615861032104203 21A91230240123015021065012309803210560120510321043023529 4
123065032103602305470532034012301502106501230584053210560120510320421032053012
306503210960123015021065012358320560230520123065032064023015021065012308603206
715612513243102352149061401206986021041502305608906503219412305120 1A5037023051
076207502352102960214319612036732785280236120673059306926014078012345320526A82
160152305107921841234530752034123453205275632

Coil, Dimension 12, Length 1276

012031430523041342305206230253452145041301204702503413025034603510341021450413012362145218241254352032602503243140325034130216521031437103412435241250352142634502103140541254081402143015341301295210314054125430153624125305214253421438234521450260421403105271534521450413012046025034130250342041254824125435264250310214036901526153BA351625741031423764021031405412014624130250318640325034130250423052062302534521450371304124350324125305216510314052862412543520326541021430153413021952103140541254301536241253052142534214382345214502604214031052715345214504130120460250341302503420412548241254352642503102140369305 2A76421450413012045362412530521425345724502640216740396304120462054125435289523452145026042140310527152410340541250492041206420541254320924052145235 03B4052103BA3012508798052013AB3102503674BA4723102503AB3052019B32630520159205264206103546350234051 2B0367BA723A923872067037839716845025785205601203278243 7B362BA260A24120643B46234BA436501AB109B291526531602463752765069540210 78A1B

9.3 Symmetrical Coils

Note that half of these transition sequences are provided.

Symmetric Coil, Dimension 9, Length 186

012341254105435021403641543502104534023412357315412341043541523451026410543502104354014321078

Symmetric Coil, Dimension 10, Length 362

012340325031506512304325431540152310457340312513415045243016043012514051305230432168123403250312340312514751230432543154015231036134032504732430452154315031203402701521340385041321 9

Symmetric Coil, Dimension 11, Length 662

012031042054012403604217420310428724013021981203104210A90124013021742031042034904203104 20A401240360428620310458401302403A90342031048540130243 90A9203104284502406304201821302439406340260 3A153106201520628024063421891065760198124360420862034903104269A430243103563026802406342180 6A962481042091208126813680421840692860A302403104A7602687A

Symmetric Coil, Dimension 12, Length 1222

012342105216201257652108701259852 10A12590521BA12509523025A125B128321781256752 1BA1257652147AB125BA5842103B52162012382032576523016521681248256752102612301830783652 36A328423A6325985236A3259652375B123756123A632569521A7123B419814B3217A124821A7123710523A6325710758457017523 6A32506523708732596523A63210570A32501732492371B7943254973842573243954237561256852461064573249578417324037087 5B712863A7123A7B07302189125842562752463 4BA17521A714823A17521A26598542347A3B

9673214265214283217841034832167362852 19B16A71AB10762701B5289750A58423A426A1
25028571A72B38245AB71A72A5671245B617250620B972692A72365425730142812 5A85238
03B85BA6381B

Symmetric Coil, Dimension 13, Length 2354

012345231032631237632187123983 21A9123BA321CB014325431034730123673210860123
983210A90123452310329A230132892310327823013267143 01B0376326B1086143983210A90
1236732109A012389321B0614398328B10A901239834B90354310347301236732108601239 83
21C910B234123913893210680123763210 36C30134631032931236734921032938912376321 03
701984108390129A216813483278210860138032103674860B30680123763210374301 34C234B
238431068012376321037489301270326732130843013763103263 10C21378127623103623 1B6
236728476310326308637684308613C032178123673213480341C8635218607306832 63C4367
308603A530680376305C0367308609A0680376301A5103673086396873860362146 9A3103670
873128376C123680376301326728417910657094 82A5219549A230AC318410A6014827608946
27A82762C52672860917C06701482760C26721706315A01AC306703CA132768523016801289
085634607280396410A18264518614692584682186370 87A073016A82652108635C306801586
9468218A5601469A6398678C0310970123456309106801543 8A03261063708A0760182562170
B8760315A10A93670586936850A372B0680B2948623087306931692730791 3A946093801849
175236801843062943268230728430549348370190B837B80127812530635801946812 3612A39
23162A14A2847327C4810A3063190632A927BA074316AB3067A976037968076012634860 76
5743075091784395083 0A1C381086039B8273B

ACKNOWLEDGMENTS

We thank Dustin Cline and Thomas Horton for their important contributions to this project, Snake-in-the-Box research, and the Institute for Artificial Intelligence. This study was partially supported by resources and technical expertise from the University of Georgia Computer Science Department. This study was also partially supported by resources and technical expertise from the Georgia Advanced Computing Resource Center (GACRC), a partnership between the University of Georgia's Office of the Vice President for Research and Office of the Vice President for Information Technology.

REFERENCES

[1] W. Potter: Snake-In-The-Box Dictionary (2014) [Online]. Available: http://ai1.ai.uga.edu/sib/sibwiki/doku.php/dictionary. Accessed 2014, Jun. 5.

[2] W. Kautz: Unit-distance Error-Checking Codes. IRE Trans. Electron. Comput, vol. EC-7, no. 2, pp. 179–180, 1958.

[3] V. Klee: What is the Maximum Length of a d-Dimensional Snake?. In: The Amer. Math. Monthly, vol. 77, no. 1, pp. 63–65, 1970.

[4] A. Hiltgen and K. Paterson: Single Track Circuit Codes. In: IEEE Trans. on Inform. Theory, vol. 47, pp. 2587–2595, 2000.

[5] F. Zhang and H. Zhu: Determination of optimal period of absolute encoders with single track cyclic gray code, in: J. of Central South University of Technology, vol. 15, no. 2 Suppl., pp. 362–366, 2008.

[6] M. Blaum and T. Etzion, "Use of snake-in-the-box codes for reliable identification of tracks in servo fields of a disk drive," U.S. Patent 6 496 312, December 17, 2002.

[7] Y. Yehezkeally and M. Schwartz: Snake-in-the-Box Codes for Rank Modulation. In: IEEE Trans. Information Theory, vol. 58, no. 8, pp. 5471–5483, 2012.

[8] I. Zinovik, Y. Chebiryak and D. Kroening: Periodic Orbits and Equilibria in Glass Models for Gene Regulatory Networks. In: IEEE Trans. Information Theory, vol. 56, no. 2, pp. 805–820, 2010.

[9] D. Davies: Longest ‘separated’ paths and loops in an N cube. In: IEEE Trans. Electron. Comput, vol. 14, p. 261, 1965.

[10] W. Potter, J. Robinson, J. Miller and K. Kochut: Using the Genetic Algorithm to Find Snake- In-The-Box Codes. In Proc. 7th Int. Conf. Industrial & Engineering Applications of Artificial Intelligence and Expert Systems, Austin, TX, 1994, pp. 421–426.

[11] K. Kochut: Snake-In-The-Box Codes for Dimension 7. In: J. Combinatorial Math. and Combinatorial Computing, vol. 20, pp. 175–185, 1996.

[12] D. Kinny: A New Approach to the Snake-In-The-Box Problem. In Proc. 20th European Conf. Artificial Intelligence, ECAI-2012, Montpellier, France, 2012 © The Author. doi: 10.3233/978-1-61499-098-7-462

[13] B. Carlson and D. Hougen: Phenotype feedback genetic algorithm operators for heuristic encoding of snakes within hypercubes. In: Proc. 12th Annu. Genetic and Evolutionary Computation Conf., GECCO '10, Portland, Oregon, 2010, 791–798.

[14] D. Kinny: Monte-Carlo Search for Snakes and Coils. In: Proc. 6th Multi-Disciplinary Int. Workshop on Artificial Intelligence, MIWAI-2012, Ho Chin Minh City, Vietnam, 2012, 271–283.

[15] P. Brooks: Particle Swarm Optimization and Priority Representation: M.S. thesis, Artificial Intelligence, Univ. Georgia, Athens, 2012.

[16] D. Casella and W. Potter: Using Evolutionary Techniques to Hunt for Snakes and Coils. In: Proc. IEEE Congress on Evolutionary Computation, CEC '05, Edinburgh, Scotland, 2005, pp. 2499–2505.

[17] M. Juric, W. Potter and M. Plaskin: Using the Parallel Virtual Machine for Hunting Snake-in-the-Box Codes. In: Proc. 7th Conf. North American Transputer Research and Applications Conf., NATUG-7, Athens, GA, 1994, 97–102.

[18] J. Bishop: Investigating the snake-in-the-box problem with neuroevolution. Dept. of Computer Science, Univ. Texas, Austin, 2006.

[19] L. Adelson, R. Alter and T. Curtz: Long Snakes and a Characterization of Maximal Snakes on the d-cube. In Proc. 4th SouthEastern Conf. Combinatorics, Graph Theory, and Computing, Congr. No. 8, Boca Raton, FL, 1973, pp. 111–124.

[20] H. Abbott and M. Katchalski: On the Construction of Snake in the Box Codes. Utilitas Mathematica, vol. 40, pp. 97-116, 1991.

[21] E. Wynn: Constructing circuit codes by permuting initial sequences. In: Proc. 20th European Conf. Artificial Intelligence, ECAI-2012, Montpellier, France, 2012 © The Author. doi: 10.3233/978-1-61499-098-7-468 [Online]. Available: http://a{}rxiv.org/abs/1201.1647. Accessed 2014, Jun. 5.

[22] W. Potter: SIB Records (2014) [Online]. Available: http://ai.uga.edu/sib/sibwiki/doku.php/records. Accessed 2014, Jun. 5.

[23] S. Meyerson, T. Drapela, W. Whiteside and W. Potter: Finding longest paths in hypercubes, snakes and coils. In Proc. 2014 IEEE Symp. Computational Intelligence for Engineering Solutions, CIES'14, pp. 103-109, Orlando, FL, 2014.

[24] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, New York: Prentice-Hall, 1995.

[25] D. Tuohy, W. Potter and D. Casella: "Searching for Snake-in-the-Box Codes with Evolved Pruning Models. In Proc. 2007 Int. Conf. Genetic and Evolutionary Methods, GEM 2007, Las Vegas, NV, 2007, pp. 3–9.

[26] P. Diaz-Gomez and D. Hougen: Genetic Algorithms for Hunting Snakes in Hypercubes: Fitness Function Analysis and Open Questions. In: 7th ACIS Int. Conf. Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2006, Las Vegas, NV, 2006, pp. 389–394.

CHAPTER 4

CONCLUSION

1. Recap

Our approach was largely inspired by efforts in (Klee, 1970), (Casella, 2005), (Tuohy, 2007), (Juric, 1994), and (Potter, 1994) to use a GA to improve known lower bounds on this difficult problem. The GA will usually yield good solutions to the SIB problems, and by applying more computation effort, the solutions can be improved to a point; however, the GA applied to the SIB problem runs into diminishing returns. The authors believe that it is extremely difficult to construct crossover schemes that contribute to results with a GA as applied to this problem. Carlson found a way around this, creating and prioritizing a list of construction rules (Carlson, 2010). By using the first rule in order of priority that creates a valid next additional node in the snake sequence. This list could be crossed over in a meaningful way with other such lists. Our reaction to the same issue was to avoid completely an inter-sub-solution communication function like crossover, or local best / best ever of particle swarm. Our process has no inter-solution communication, only inter-solution competition.

We use a fitness based heuristic to selects dense solutions that are highly likely to dead end, but are the only prospects for an excellent result. We generate millions of these solutions expecting the vast majority of them to fail. We invest computational effort in those most likely to give exceptional results. Our process is unique in that it assesses the remaining material (nodes) in the hypercube, scoring its potential to be grown into an exceptional solution. We believe that

we have only scratched the surface on the opportunity that computationally cost effective graph theory based approaches to assessing the hypercube represent.

Increasing the beam width in our process by 50–100 times netted only a small return, while small improvements in the predictive value of our fitness function resulted in new lower bounds in several dimensions and solution classes (snakes / coils / symmetrical coils).

Exploiting the inherent constraints of specific solution classes (particularly coils and symmetrical coils) in the Snake-in-the-Box problem enabled us to dramatically reduce the effective search space. Once the space is reduced, then we apply heuristics to select from a still intractable number of solutions. Imposing powerful new validity based checks on the combinatorial explosion of sub-solutions led directly to lower bounds in many dimensions and classes of the problem. For example, the symmetrical coil aspect of the SIB problem is the most constraining. We were able to assess whether a sub-solutions could reach coil-forming end points and thus retain coil forming potential. If not we discarded the sub-solution. We also only created sub-solutions that adhered to the symmetrical property of symmetric coils. This left us with a small fraction of the search space that we would have had to traverse without the intelligent application of known constraints.

These innovations explain the particular strengths of our process when applied to coils and symmetrical coils; symmetrical coils which are the most constrained by definition. We are particularly pleased with the gains achieved in dimensions 9–13, which saw lower bounds increases of 6, 12, 22, 94, and 456 nodes respectively.

In general, beam search is an excellent platform to generate strictly valid sub-solution where the problem representation allows for valid incremental additions to a sub-solution. Beam search

allowed us to separate valid solution generation from fitness / selection. Once we had an engine

to generate strictly valid sub-solutions that adhered to Kochut's Constraint, we focused our efforts entirely on processing improvements — memory efficiency and parallelism — and improving the predictive value of our fitness calculations.

Increasing the number of solutions considered (beam width) by orders of magnitude led to diminishing improvement, while seemingly small adjustments in the way fitness was calculated led to large to jumps in performance.

2 Future

If we can find and apply enough valid constraints to these problems, while increasing the amount of computation force applied, perhaps we can exhaustively explore dimensions eight or greater. On the way to constraining the problem enough to make exhaustive search feasible we have found such constraints predictably improve the performance of our heuristic and probably others, resulting in the improved lower bounds published here.

In future work we plan to explore more graph theory based improvements. One of the most promising is articulation point detection algorithms. An articulation point detection algorithm efficiently, O (V+E), locates nodes of an undirected graph, in our case the remaining valid reachable nodes of a hypercube, where removal of a single node will break the graph into at least two unconnected parts. Since articulation point detection will also identify blind alleys and dead end nodes as they are special cases of articulation points, the current expense of blind alley detection would be eliminated. An efficient implementation of articulation point detection will be a net computational expense reduction versus our current process.

Again, an articulation point is a point that if removed will break a graph into two or more unconnected sub-graphs. This is of particular interest to our problem because each articulation

point found represents an *exclusive* fork in the possible paths of a snake or coil. A sub-solution may pass through an articulation point to a sub-graph defined by this point, but may never return to its previous subgraph or any other subgraph defined by this articulation point because of the a-chordal constraint. Applying this insight to our current fitness calculation process, which is strongly based on remaining reachable nodes in the hypercube, we would remove all the nodes in the smaller sub graph(s) defined by an articulation point. In other words, we would eliminate all subgraph nodes defined by this point, except for the largest sub-graph, for the purpose of determining fitness.

This approach can be even more powerful when applied to coils and symmetrical coils. As we have discussed, coils and symmetrical coils have known end points. If any of the sub-graphs delineated by an articulation point does not contain an end point, then it cannot be part of the eventual coil or symmetrical coil solution. These sub-graphs nodes can be eliminated from the reachable available node count as they can never be part of a valid coil solution.

Since a snake or coil sub-solution cannot be extended any longer than the remaining reachable available nodes left in the hypercube, implementing articulation point detection should not only improve the predictive value of fitness, but may make exhaustive calculation of longest snakes, coils, and symmetrical coils feasible.

Another possible approach that includes articulation point detection would be to search for snakes using defined end points. This would include techniques like our symmetric coil search, where multiple sub-solutions with many different end points are considered within the same search; allowing competition among solutions based on end points. This might also include an orderly search by start point (node 0) and by all other valid end points in the hypercube, each taken one at a time, further constraining the search.

A final related approach would be to use the symmetrical coil finding approach without a symmetry constraint. Applying articulation point detection to this approach would improve this as well, though this would involve additional complexity in implementation where we previously relied on the symmetry constraint.

3 A Different Tack - Mathematical Construction / Abbott Acknowledgment

Abbott set many record lower bounds for the problem and whose last standing record; a coil in dimension 13 was set in 1991 (Abbott, 1991), has stood for 22 years. We were confident that we would re-set most lower bounds for snakes, coils, and symmetrical coils for dimensions 10-13 and certainly thought we would set new lower bounds for the longer standing current high dimension lower bounds. For the most part his was true, but despite vastly superior hardware, and the benefits of Abbott's and other's research, we were not able to improve upon Abbott's record coil of length 2468 in dimension 13. We applied our process to the dimension 13 coil problem several times; finally using nearly 23,000 hours of 2 GHZ core time. We tied Abbott's lower bound for coils in dimension 13 twice, but were unable to surpass it. Perhaps we should include a bias in future work, one of the primary assumptions of a mathematical approach, Abbotts's approach; many of the best solutions are orderly.

References

H. Abbott and M. Katchalski, "On the Construction of Snake in the Box Codes," *Utilitas Mathematica*, vol. 40, pp. 97-116, 1991.

L. Adelson, R. Alter and T. Curtz, "Long Snakes and a Characterization of Maximal Snakes on the d-cube," in *Proc. 4th SouthEastern Conf. Combinatorics, Graph Theory, and Computing, Congr. No. 8*, Boca Raton, FL, 1973, pp. 111–124.

J. Bishop, "Investigating the snake-in-the-box problem with neuroevolution," Dept. of Computer Science, Univ. Texas, Austin, 2006.

M. Blaum and T. Etzion, "Use of snake-in-the-box codes for reliable identification of tracks in servo fields of a disk drive," U.S. Patent 6 496 312, December 17, 2002.

P. Brooks, "Particle Swarm Optimization and Priority Representation," M.S. thesis, Artificial Intelligence, Univ. Georgia, Athens, 2012.

B. Carlson and D. Hougen, "Phenotype feedback genetic algorithm operators for heuristic encoding of snakes within hypercubes," in *Proc. 12th Annu. Genetic and Evolutionary Computation Conf., GECCO '10*, Portland, Oregon, 2010, 791–798.

D. Casella and W. Potter, "Using Evolutionary Techniques to Hunt for Snakes and Coils," in *Proc. IEEE Congr. on Evolutionary Computation, CEC '05*, Edinburgh, Scotland, 2005, pp. 2499–2505.

D. Davies, "Longest 'separated' paths and loops in an N cube," *IEEE Trans. Electron. Comput.*, vol. 14, p. 261, 1965.

M. Garey and D. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, San Francisco: Freeman, 1979.

A. Hiltgen and K. Paterson, "Single Track Circuit Codes," *IEEE Trans. on Inform. Theory*, vol. 47, pp. 2587–2595, 2000.

M. Juric, W. Potter and M. Plaskin, "Using the Parallel Virtual Machine for Hunting Snake-in-the-Box Codes," in *Proc. 7th Conf. North American Transputer Research and Applications Conf., NATUG-7*, Athens, GA, 1994, 97–102.

W. Kautz, "Unit-distance Error-Checking Codes," *IRE Trans. Electron. Comput.*, vol. EC-7, no. 2, pp. 179–180, 1958.

D. Kinny, "A New Approach to the Snake-In-The-Box Problem," in *Proc. 20th European Conf. Artificial Intelligence, ECAI-2012*, Montpellier, France, 2012 © The Author. doi: 10.3233/978-1-61499-098-7-462

V. Klee, "What is the Maximum Length of a d-Dimensional Snake?" *The Amer. Math. Monthly*, vol. 77, no. 1, pp. 63–65, 1970.

K. Kochut, "Snake-In-The-Box Codes for Dimension 7," *J. Combinatorial Math. and Combinatorial Computing*, vol. 20, pp. 175–185, 1996.

W. Potter. *Snake-In-The-Box Dictionary* (2014) [Online]. Available: http://ai1.ai.uga.edu/sib/sibwiki/doku.php/dictionary. Accessed 2014, Jun. 5.

W. Potter, J. Robinson, J. Miller and K. Kochut, "Using the Genetic Algorithm to Find Snake-In-The-Box Codes," in *Proc. 7th Int. Conf. Industrial & Engineering Applications of Artificial Intelligence and Expert Systems*, Austin, TX, 1994, pp. 421–426.

Y. Yehezkeally and M. Schwartz, "Snake-in-the-Box Codes for Rank Modulation," *IEEE Trans. Information Theory*, vol. 58, no. 8, pp. 5471–5483, 2012.

I. Zinovik, Y. Chebiryak and D. Kroening, "Periodic Orbits and Equilibria in Glass Models for Gene Regulatory Networks," *IEEE Trans. Information Theory*, vol. 56, no. 2, pp. 805–820, 2010.

F. Zhang and H. Zhu, "Determination of optimal period of absolute encoders with single track cyclic gray code," *J. of Central South University of Technology*, vol. 15, no. 2 Suppl., pp. 362–366, 2008.