New Lower Bounds for the Snake-in-the-Box Problem: Using Evolutionary Techniques to Hunt for Snakes

D. A. Casella and W. D. Potter

Artificial Intelligence Center
111 GSRC, Univ. of Georgia
Athens, Georgia 30602-7415
artdeco@uga.edu
potter@uga.edu

Abstract

The snake-in-the-box problem is a difficult problem in mathematics and computer science that was first described by Kautz in the late 1950's (Kautz 1958). Snake-in-the-box codes have many applications in electrical engineering, coding theory, and computer network topologies. Generally, the longer the snake for a given dimension, the more useful it is in these applications (Klee 1970). By applying a relatively recent evolutionary search algorithm known as a population-based stochastic hill-climber, new lower bounds were achieved for the longest snake in each of the dimensions nine through twelve and the longest coil in each of the dimensions nine through eleven.

Introduction

Hunting for 'snakes,' or achordal induced paths in an n-dimensional hypercube deals with finding the longest path, following the edges of the hypercube, that obeys certain constraints. First, every two nodes, or vertices that are consecutive in the path must be adjacent to each other in the hypercube, and second, every two nodes that are not consecutive in the path must not be adjacent to each other in the hypercube. A third constraint, whether the first node in the path is adjacent or non-adjacent to the last node in the path, determines if the path is a 'coil' or a 'snake.' While coils have received the most attention in the literature (Harary, Hayes, and Wu 1988), both snakes and coils have useful applications. This paper deals primarily with searching for open paths, or snakes, where the start node is not adjacent to the end node.

An n-dimensional hypercube contains 2^n nodes that can be represented by the 2^n n-tuples of binary digits of the hypercube's Gray code. By labeling each node of the hypercube with its Gray code representation, adjacencies between nodes can be detected by the fact that their binary representations differ by exactly one coordinate. Using this fact, one can immediately detect if any two nodes in the hypercube are adjacent by performing an exclusive-or, hereafter referred to as XOR, operation on them and confirming that the result is an integer power of two. Using this information, a strategy for detecting, as well as generating, node adjacencies can be generalized to any dimension. In Figure 1, a path through the three-dimensional hypercube is highlighted. This path, {0, 1, 3, 7, 6} in integer representation and {000, 001, 011, 111, 110} in binary representation, is one specific example of a longest-possible snake for a three-dimensional hypercube. The length of this snake is four, as the length of a snake or coil always refers to the number of transitions, or edges, in the path.

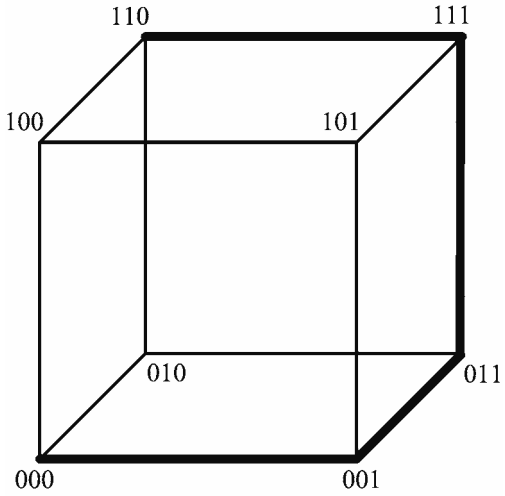

Figure 1: A Three-Dimension Hypercube with Embedded Snake

Approaches

Traditionally, mathematical approaches to the snake-in-the-box, hereafter referred to as SIB, problem have involved two fundamental strategies. The first is a method of construction utilizing the tools of logic, discrete mathematics, and graph theory, while the second strategy uses mathematical analysis to reduce the search-space followed by a computationally exhaustive search of the remaining, tractable, search-space. These techniques have been successful in determining the longest-possible snakes for hypercubes of dimensions one through seven as shown in Table 1. For dimension seven, the lower bound for longest-possible snake (Potter et. al. 1994) was found independently using both a genetic algorithm and an exhaustive search, while the lower bound for longest-possible coil (Kochut 1996) was found using only an exhaustive search.

DIM	SNAKE	COIL
1	1	0
2	2	4
3	4	6
4	7	8
5	13	14
6	26	26
7	50	48

Table 1: Maximum Lengths of Snakes and Coils

But even as mathematical approaches to solving this problem have been strengthened through the use of computational techniques, the combinatorial explosion in the size of the search-space as the dimension number increases has become a barrier for these methods. For dimensions eight and above, even with currently available hardware, an exhaustive search remains impractical. This has opened the door for less-traditional, heuristic-based computational search techniques. One increasingly popular branch of these search techniques is known as *stochastic search algorithms.* This area includes stochastic hill-climbers, tabu search, simulated annealing, evolutionary strategies, genetic algorithms, and hybrids of these particular types such as memetic algorithms.

One example of a stochastic search algorithm that has been used to hunt for snakes in dimension eight is the genetic algorithm, hereafter referred to as GA. Developed by John Holland (Holland 1975), the GA is based on the simulation of Darwinian evolution and uses an evolutionary loop composed of fitness-based selection of individuals from a population, crossover of these individuals' genetic material, and mutation of these individuals' genes. The GA performs a search-space reduction through the use of a heuristic in determining the fitness of an individual within the population and through the inherent parallelism of a population-based approach. This technique has met with success and, by finding, what was at that time, a record-breaking snake in dimension eight (Potter et. al. 1994), proved its effectiveness for snake hunting.

Another stochastic search algorithm that has proven successful in traveling salesperson-type problems, such as the SIB problem, is the stochastic hill-climber (Kingdon and Dekker 1995). A population-based stochastic hill-climber, hereafter referred to as PBSHC, is very similar in structure and operation to the simple GA with a few key differences in the evolutionary cycle. The first difference is the absence of a crossover operator. Where the GA models sexual reproduction, the PBSHC models asexual reproduction in that the children in each generation are created directly from the parents of the previous generation without the exchange of genetic material between them. The second difference is the addition of a growth operator. The growth operator is the component that does the actual 'hill-climbing' as it chooses randomly from the nodes available to extend each snake's path by one edge along the hypercube. The absence of crossover, broadly considered the most important operator of a GA, can sometimes leave the PBSHC at a relative disadvantage. However, due to the trouble most crossover schemes have in dealing with adjacencies, the lack of crossover did not seriously degrade the PBSHC's performance, relative to the GA, in the case of the SIB problem.

The Evolutionary Cycle

Each individual in the population consists of a sequence of integers that represents the node sequence of a snake, or valid path through the hypercube, in the dimension being searched. These individuals are initialized as either a snake of length zero, that is consisting of only the zero node, or seeded with a pre-existing snake of choice. Following initialization, the evolutionary cycle begins its first generation. Each generation begins with a fitness evaluation. The fitness function used is based on both the length of the snake and the 'tightness' of the snake. The tightness of a snake is a measure of how many nodes are left available in the hypercube after subtracting all those

nodes that are disqualified either by already being in the snake or by being adjacent to a node, other than the start node or the end node, that is already in the snake. The choice of tightness as a component of the fitness function was inspired by the idea that tighter snakes might tend to be longer snakes. Since all snakes that were able to grow in the previous generation have the same length, tightness becomes the only distinguishing factor of the fitness function.

After the individual fitness of each of the snakes in the population is determined, the population is subjected to selection based upon each snake's fitness. Three fundamental selection types were considered. Preliminary trials were conducted using roulette-wheel, tournament, and rank-based selection methods. Roulette-wheel, or probabilistic selection, proved the least effective, and most computationally expensive. Tournament selection was more effective than roulette-wheel at producing longer snakes, on average, over multiple runs. Rank-based selection out-performed both roulette-wheel and tournament selection, by maintaining a more diverse population throughout the evolutionary cycle. In rank-based selection, after first ranking the population by fitness, selection takes place based on a set percentage of the population.

After selection, the growth operator grows each snake in the population by one step each generation, connecting each snake's end node to one of its adjacent nodes that has not been disqualified by already being in the snake, or by being adjacent to some node that is in the snake. Both unidirectional and bi-directional growth were implemented, with bi-directional growth allowing each snake to grow from either end. This operator can be seen to perform a stochastic hill-climbing process on each snake in the population as the choice of which adjacent node to connect to is based on random selection from the available nodes. A choice was made early to grow all snakes in the population instead of only the snake of best fitness (note: typically, enhancing the best individual in a population is a standard approach used in hybrid genetic algorithms (Potter et. al. 1992)). This choice was also based on results of a comparison of these two approaches in an earlier GA implementation. Growing all the individuals within the population works well in conjunction with the fitness function, but does require that all snakes in the population be of the same length in order to function properly. The fitness function consists of the sum of the snake's length and normalized tightness. This results in the fitness function simplifying to a function of tightness alone as the length component of the fitness value will dominate for snakes of different lengths, yet cancel for snakes of the same length. This also results in the automatic elimination of snakes that can no longer grow, allowing their place in the population to be reallocated to other snakes that are still capable of growth.

After growth, the mutation operator acts on each snake in the population. The mutation operator we experimented with is an enhancement of a basic XOR mutation scheme (Brown 2004). In the standard XOR mutation scheme, a node is chosen at random from within the snake, excluding the start node and the end node. The chosen node's neighboring nodes are XOR'd and that result is then XOR'd with the chosen node in order to exchange it with a different node that maintains local adjacency requirements with the original node's neighbors. The enhancement of this operator is referred to as iterative-conditional XOR mutation and is somewhat more computationally expensive, but also more effective. Instead of choosing a node at random, each node in the snake, with the exception of the start node and the end node, is mutated and tested for any improvement in fitness within a copy of the original snake. If any improvement is found, that mutation is added to a mutation pool. Upon testing of all nodes within the snake, the snake of best fitness from the mutation pool is substituted for the original snake. If any improvement was found through mutation, the entire process is repeated until no further improvement is found. This scheme ensures that only constructive mutation is allowed, and that undergoing mutation can never reduce an individual's fitness. By making this mutation conditional, it has also become a 'hill-climbing' component of the evolutionary cycle. However, left to iterate without bounds this scheme may lead to prohibitive runtimes. Restricting the number of iterations for each original snake to five for example keeps the advantages of a local hill-climber and also keeps the runtime under control. It turns out that this enhancement did not influence our results because the operator was turned off in order to prevent changes in the high-quality root-snakes used when seeding results from lower to higher dimensions.

PBSHC Experimental Setup

The choice of parameter settings was found to be of key importance to the performance of the PBSHC and these settings were tuned extensively in dimension eight before modified to run in dimensions nine through twelve. Our previous experience with genetic algorithm snake hunters supported the application of lower dimension parameter settings to higher dimensions. The fitness function is set to the sum of length and normalized tightness. Normalized tightness was defined as the number of nodes remaining available divided by the total number of nodes in the n-dimensional hypercube. Rank-based selection was chosen in order to maximize diversity within the population.

While both unidirectional and bi-directional growth were used in the dimension-eight trials, the relatively memory-conservative unidirectional implementation was chosen for these experiments in order to maximize potential population sizes for dimensions nine through twelve. Because this particular implementation's chromosome size is constant, the use of unidirectional growth instead of bi-directional growth allowed the required memory for each population size to be reduced by half.. Population sizes from one hundred through ten thousand were run in trials using mutation. Larger populations were prohibitively time-consuming using mutation, especially for dimensions eleven and twelve.

Results

The best results to date were achieved using populations of ten thousand, a selection percentage of ninety percent, and by seeding each population at startup. Using a technique where the best snakes found in each dimension were used as seeds for the next higher dimension (Potter et. al. 1994), the PBSHC was able to find snakes longer than the previously known longest snakes for dimensions nine through twelve. However, to generate good seeds for dimension nine, a bootstrap solution was used. This method involved cutting a dimension-eight, length-97 snake (Rajan and Shende 1999) back to its length-50 root, corresponding to the longest snake in dimension seven, and running an exhaustive search on that snake to generate a pool of seventeen distinct length-97 snakes. These snakes were then used to seed the dimension-nine runs. Subsequently, the best snakes found in each dimension were used as seeds for the next dimension's runs. In conjunction with this technique, mutation was shut off in order to preserve the high-quality seeds generated from each previous dimension. This also reduced the runtime allowing larger populations to be run over a much shorter time-frame.

DIM	PREVIOUS BEST	PBSHC
8	97	97
9	168	186
10	338	358
11	618	680
12	1236	1260

Table 2: Comparison of Best-Known Lower Bounds for Snakes

In Table 2, for dimension eight, the lower bound for longest-known snake (Rajan and Shende 1999) was found using traditional mathematical proof and construction techniques aided by computational search (i.e., construction based on lower dimension long snake characteristics). For dimensions nine through twelve, the previous best-known lower bounds were derived by calculation from the lower bounds for longest-known coils in each dimension. Any coil can be converted to a snake by removing one node. This results in a snake whose length is two less than the original coil, as removing one node removes the two edges connecting it within the coil. The previous best-known lower bounds were 168 for dimension nine (Abbott and Katchalski 1991), and 338, 618, and 1236 for dimensions ten through twelve, respectively (Paterson and Tuliani 1998). The transition sequences representing the new lower bounds for longest-known snakes in dimensions nine through twelve are listed here.

dim 9 : 186

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

dim 10 : 358

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

dim 11 : 680

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

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

dim 12 : 1260

0 1 2 3 4 5 3 2 1 0 3 2 5 3 1 2 3 4 5 2 3 1 0 3 2 6 0 5 4 3 0 1
3 5 0 2 3 1 0 3 5 4 3 0 1 2 5 0 2 3 7 2 0 5 4 2 1 3 0 1 5 2 0 1
3 0 5 4 3 5 2 0 3 6 2 5 4 3 0 1 3 5 0 2 3 1 0 3 5 4 3 0 1 2 5 0
2 8 7 3 2 4 5 3 1 0 3 4 5 3 1 5 2 1 0 3 4 5 3 0 1 3 5 6 1 3 0 1
5 2 0 1 3 0 5 4 3 5 2 0 3 1 7 6 2 5 4 3 5 1 3 2 5 3 0 1 3 5 6 2
5 4 3 5 0 1 3 5 4 3 0 5 2 0 1 5 4 1 0 3 5 6 2 3 5 0 9 3 5 2 6 3
0 2 5 3 4 5 0 3 1 0 2 5 1 0 3 1 5 6 2 0 5 3 1 5 2 3 5 4 3 1 0 4
7 1 3 5 2 3 1 0 2 3 5 4 0 8 3 1 0 3 4 5 3 0 1 3 2 0 5 3 1 0 3 4
5 3 1 0 2 3 7 2 0 5 4 2 0 3 5 2 0 1 3 0 5 4 3 5 2 0 1 6 4 1 0 2
5 1 3 5 2 7 3 5 1 0 3 1 2 3 5 1 3 0 5 4 3 5 2 0 3 1 2 8 4 0 5 4
3 2 1 0 3 2 4 3 0 5 2 0 1 4 6 1 0 2 3 6 2 1 7 5 8 1 5 2 0 3 5 1
8 3 10 9 4 2 1 3 0 1 5 2 0 1 3 0 5 4 3 5 2 0 3 6 2 5 4 3 0 1 3
5 0 2 3 1 0 3 5 4 3 0 1 2 5 0 7 4 1 0 3 2 5 3 1 2 3 4 5 2 3 1 0
3 2 6 0 5 4 3 0 1 3 5 0 2 3 1 0 3 5 4 3 0 1 2 5 9 4 1 3 0 5 4 3
5 2 0 3 6 2 5 4 3 0 1 3 5 0 2 3 1 0 3 5 4 3 0 1 2 8 9 2 1 0 3 4
5 3 0 1 3 2 0 5 3 1 0 3 4 5 2 6 3 0 2 5 3 4 5 0 3 1 0 2 5 1 0 3
1 2 4 5 0 9 6 0 2 3 1 0 2 5 3 4 5 0 3 1 0 2 3 5 6 2 3 1 5 2 3 4
5 3 0 1 3 8 4 1 3 0 1 5 3 7 2 5 3 1 5 2 0 1 5 4 1 2 5 3 0 2 6 3
2 5 3 4 5 0 3 1 0 2 5 1 0 3 4 1 0 8 1 4 5 3 2 5 1 3 5 0 2 3 5 4
1 2 9 0 3 4 5 0 3 1 0 2 5 1 0 3 1 2 4 5 2 3 1 0 3 6 2 0 5 4 3 0
5 2 0 3 1 2 0 5 4 9 1 4 5 0 3 1 5 3 2 1 3 0 1 4 9 0 2 5 0 9 1 2
0 3 6 2 0 11 5 0 9 4 3 0 5 10 6 5 0 3 4 5 0 6 1 7 4 0 2 3 1 0
3 5 4 3 0 1 2 5 1 3 5 4 3 0 1 2 6 1 0 3 1 2 3 5 1 3 0 5 4 3 5 2
0 3 1 2 8 1 5 4 3 0 5 2 0 1 5 6 1 3 5 2 3 1 0 2 7 6 2 0 1 3 0 5
4 3 5 2 0 3 6 2 5 4 3 0 1 3 5 0 2 6 10 1 3 0 2 5 0 3 4 5 0 2 6
3 2 5 3 4 5 0 3 1 5 3 4 5 0 7 4 0 2 3 1 0 3 5 4 3 0 1 2 0 4 1 0
3 4 5 2 6 3 0 2 5 3 4 5 0 3 1 5 3 2 1 3 0 1 5 3 10 2 5 4 6 1 5
3 2 0 1 3 5 6 7 2 4 9 10 2 1 3 0 1 5 2 0 1 3 0 5 4 3 5 2 0 3 6
2 3 5 2 0 3 4 5 3 2 5 1 3 5 0 2 3 5 7 1 3 5 2 0 5 4 3 5 2 0 1 2
4 0 2 3 5 6 2 3 1 5 2 0 1 3 4 10 1 3 0 2 5 3 4 5 0 3 1 0 2 5 1
0 3 1 6 5 3 1 0 3 5 4 3 0 1 2 0 4 1 0 3 4 5 1 8 2 3 1 0 2 5 3 4
5 0 3 1 0 2 3 5 6 2 3 1 5 2 0 3 1 7 6 2 0 5 4 8 1 4 5 0 2 5 3 0
1 3 5 6 1 3 2 5 0 4 3 5 0 1 9 2 6 5 1 3 0 1 2 5 0 9 4 0 5 8 6 5
1 3 0 5 4 3 5 2 0 3 6 2 3 5 2 0 3 4 5 3 0 10 9 1 3 2 0 5 4 7 1
4 5 0 3 1 0 2 3 5 6 2 5 0 2 1 3 0 2 5 0 3 4 5 0 2 6 3 0 1 3 5 1
4 3 1 6 4 2 5 9 4 1 3 0 5 4 3 5 2 0 3 6 2 3 5 2 0 3 4 5 3 2 5 1
3 5 0 2 5 6 1 3 0 1 4 8 5 2 3 1 2 5 0 3 2 1 3 10 8 3 2 4 6 1 3
4 6 0 5 1 10 6 3 2 7 9 0 5 2 1 0 3 4 5 6 2 5 4 3 0 5 4 1 0 5 2
1 0 3 1 2 4 5 0 9 4 5 2 1 4 3 0 1 3 10 6 9 1 4 5 6 3 1 5 4 1 0
3 4 1 2 4 3 5 2 10 8 0 2 5 0 3 4 2 6 3 1 0 3 4

Conclusions

The results of using a PBSHC to search for SIB codes are very encouraging. As computational hardware improves over time, this technique should prove useful in even higher dimensions. Further improvements using this implementation of the PBSHC are anticipated including the return to bi-directional growth, the addition of a mid-snake growth operator, and a parallel virtual machine implementation of the PBSHC. Preliminary trials using this technique to search for coils, or closed paths, have begun and we have already discovered new lower bounds for coils in dimensions nine (180), ten (344), and eleven (630); details to be reported in a future paper.

Acknowledgements

We would like to thank the students and staff of the AI Center, especially Dr. Michael Covington, for their support and cooperation with this project. There were numerous days and nights when the Center's lab computers were occupied with snake hunting.

References

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

Brown, W. April, 2004. Personal Communication.

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

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: The University of Michigan Press.

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

Kingdon, J. and Dekker L. 1995. The Shape of Space. In *Proceedings of the Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, 543-548. London, UK.: IEE.

Klee, V. 1970. What is the Maximum Length of a d-Dimensional Snake? *American Mathematics Monthly* 77:63-65.

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

Paterson, K. G. and Tuliani, J. 1998. Some New Circuit Codes. *IEEE Transactions on Information Theory* 44(3):1305-1309.

Potter, W.D., J.A. Miller, B.E. Tonn, R.V. Gandham, and C.N. Lapena, 1992. Improving The Reliability of Heuristic Multiple Fault Diagnosis Via The Environmental Conditioning Operator. *APPLIED INTELLIGENCE: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies* 2: 5-23.

Potter, W. D.; Robinson R. W.; Miller J. A.; and Kochut, K. J. 1994. Using the Genetic Algorithm to Find Snake-In-The-Box Codes. In *7-th International Conference On Industrial & Engineering Applications Of Artificial Intelligence and Expert Systems*, 421-426. Austin, Texas.

Rajan, D. S. and Shende, A. M. 1999. Maximal and Reversible Snakes in Hypercubes. Presented at the *24th Annual Australasian Conference on Combinatorial Mathematics and Combinatorial Computation*.