Using The Genetic Algorithm to Find Snake-In-The-Box Codes

W.D. Potter [1,2], **R.W. Robinson** [1], **J.A. Miller** [1,2], **K.J. Kochut** [1,2], and **D.Z. Redys** [1,2]

[1]Department of Computer Science
415 GSRC, Univ. of Georgia
Athens, Georgia 30602-7404
potter@cs.uga.edu

[2]Artificial Intelligence Programs
111 GSRC, Univ. of Georgia
Athens, Georgia 30602-7415
dpotter@ai.uga.edu

ABSTRACT

Genetic Algorithms are heuristic search schemes based on a model of Darwinian evolution. Although not guaranteed to find the optimal solution, genetic algorithms have been shown to be effective at finding near optimal and, in some cases, optimal solutions to combinatorially explosive problems.

Finding a maximal length snake, a list of vertices satisfying specific constraints, in an n-dimensional hypercube, the "box", is the type of problem that suffers from (serious) combinatorial explosion. That is, as n increases, the size of the solution space that must be searched increases very rapidly. As a consequence, the maximum lengths of snakes have previously only been determined for $n \leq 6$.

Since their identification in the late 1950's, snakes (also known as Snake-In-The-Box codes) have received a significant amount of research attention. Snake-In-The-Box codes have a variety of applications in areas such as error-detection in analog-to-digital conversion, electronic combination locking schemes, and disjunctive normal form simplification. Given a certain dimension for a hypercube, the longer the snake or list of codewords, the more useful it is in the application.

In this paper, we discuss experiments conducted to find snakes in hypercubes of dimension 7 and 8. To date, we have broken the previous world record for the lower bound in dimension 7 and in dimension 8 using our Genetic Algorithm approach.

INTRODUCTION

Genetic algorithms are heuristic search methods based on the notion of the survival of the fittest. They have been applied to a wide variety of problems, for example: Multiple Fault Diagnosis (MFD) [Davi91, Mill93, Pott90a, Pott90b, Pott92a]; Set Covering (SC) and Traveling Salesman Problems (TSP) [Liep90]; communication network configuration [Pott92b]; and control of natural gas pipelines and game playing [Gold89]. Genetic algorithms are a type of stochastic search method, and are applied to NP-Hard problems in many areas ranging from scheduling optimization to designing optimal configurations and layouts. Developed by John Holland in 1975 [Holl75], they are modeled from and mimic the theory of natural evolution.

In genetic algorithms, a population is simply a collection of "chromosomes" representing possible solutions to the specific problem at hand. These chromosomes are altered or modified using the genetic operators in order to create a new generation. This evolutionary process is repeated a predetermined number of times or until no improvement in the solution to the problem is found.

Encoding Schemes

Originally, the chromosomes (or individuals) in the population were represented as strings of binary digits. Later, other types of representations were experimented with and found to be, depending on the problem to be solved, more convenient. However, bit string representations are still the most commonly used encoding techniques and have been used in many real-world applications of genetic algorithms. Such representations have several advantages: they are simple to create and manipulate, many types of information can be easily encoded, and the genetic operators are easy to apply [Davi91]. Other types of encodings include real number representations (e.g., applied to parametric design of aircraft [Davi91], and network design [Pott92b]), and ordered list representations (e.g., applied to scheduling optimization [Davi91], traveling salesman problems, and our own hypercube path finding research).

Evaluation

The evaluation of a chromosome is done to test its "fitness" as a solution, and is achieved, typically, by making use of a mathematical formula known as an objective function (non-mathematical approaches have also been used). The objective function plays the role of the environment in natural evolution by rating individuals in terms of their fitness. Choosing and formulating an appropriate objective function is crucial to the efficient solution of any given genetic algorithm problem. In our case, searching for long paths, a fitness based on the length of the path found so far is a very good function.

Genetic Operators

Genetic operators are used to alter the composition of chromosomes. The fundamental genetic operators, mate selection, crossover and mutation, are used to create children (or individuals in the next generation) that differ from their parents (or individuals in the previous generation). Additional advanced genetic operators have been inspired by knowledge derived from the field of genetics (e.g., inversion dominance, diploidy, and abeyance) [Gold89].

Using the mate selection operator, individual chromosomes are selected according to their fitness which is evaluated using the objective function. This means that a chromosome with a higher fitness value will have a higher probability of contributing one or more offspring to the next generation. There are many ways this operator can be implemented. A basic method calls for using a weighted roulette wheel with slots sized according to fitness [Gold89]. Essentially, the probability of an individual being selected is proportional to the individual's fitness. Individuals thus selected are further operated on by the other genetic operators: crossover and mutation.

The purpose of the crossover operator is to produce new chromosomes that are distinctly different from their parents, yet retain some of their parents characteristics. There are two basic crossover techniques, called one-point crossover and two-point crossover. In one-point crossover, two parent chromosomes are interchanged at a randomly selected point thus creating two children. In two-point crossover, two crossover points are selected instead of just one crossover point. The part of the chromosome string between these two points is then swapped to generate two children. Empirical studies have shown that two-point crossover usually provides better randomization than one-point crossover. Other crossover techniques such as uniform crossover are considered whenever it is found that both one-point and two-point crossover techniques are not combining useful characteristics of chromosomes from the parents. In uniform crossover, for each bit position within the new child chromosome, it is decided randomly which parent the child will inherit the bit from. As discussed in [Davi91], it has been found that uniform crossover is inferior to two-point crossover in certain instances. For other representation schemes, specialized crossover techniques have been introduced. Several examples include order crossover, edge recombination, and partially matched crossover [Davi91, Gold89].

Some of the individuals in the new generation produced by mate selection and crossover are mutated using the mutation operator. The most common form of mutation is to take a bit from a chromosome and alter (i.e., flip) it with some predetermined probability. As mutation rates are very small in natural evolution, the probability with which the mutation operator is applied is set to a very low value and is generally experimented with before this value is fixed.

SNAKES AND COILS

An *n-coil* in Q_n, the n-dimensional unit cube, is a simple cycle C in Q_n such that C has no chords in Q_n. That is, every edge of Q_n which joins two nodes of C is in fact an edge of C. The nodes of Q_n are the 2^n n-tuples of binary digits, and two nodes are joined by an edge of Q_n if they differ in exactly one coordinate. An *n-snake* is a simple (open) path S in Q_n which has no chords in Q_n. In other words, an n-snake is a path in Q_n having adjacencies only between nodes which are consecutive in the path. In this paper, we refer (following [Hara88]) to an n-coil and n-snake as simply a *coil* and *snake*, respectively. Figure 1 shows an example of Q_3. The binary labels for Q_3 presented in a Gray code ordering are:

000 001 010 011 100 101 110 111

Using the binary labels, adjacent nodes differ by only 1 bit. For example, 010 is adjacent to 000, 011, and 110. A Gray code is a Hamiltonian path or cycle in the hypercube. For dimension $n \geq 3$, every Gray code contains at least one chord. In the Gray code above, for instance, 000 and 100 are adjacent in Q_3 but not in the path, so this edge is a chord. Figure 1 also shows a maximum length coil in Q_3 containing 6 edges connecting nodes:

000 001 011 111 110 100 000

Figure 2 shows a maximum length coil in Q_4 which contains 8 edges. The *length* of a coil or a snake is always the number of edges.

Hypercubes have long been studied for their relevance to coding theory [Kaut58], and more recently due to the advent of parallel computing systems with hypercube communication topologies; see [Hara88] for a survey of known results on hypercubes. Coils (or closed snakes) in hypercubes have received the most attention in the literature [Abbo88, Hara88]. Whether open or closed, snakes in Q_n have various applications, such as error-detection in analog-to-digital conversion [Klee70]. In this case, the longer the snake, the more accurate the conver-

Figure 1.

Figure 2.

sion will be. Additionally, snakes are related to algorithms used for disjunctive normal form simplification and for electronic combination locking schemes; again, the longer the snake, the more useful it is in the application [Klee70]. Finally, the length of the longest snakes corresponds to the worst-case number of iterations in local-search algorithms [Tove81]. Consequently, great interest has developed in determining the maximum length of open snakes, s_n, and coils (closed snakes), c_n, among all snakes in Q_n; see for example [Kaut58, Davi65, Sing66, Danz67, Klee67, Doug69, Adel73, Deim85, Abbo88, Wojc89, Abbo91a, Snev93]. However, finding long snakes is such a difficult problem that the maximum lengths of snakes were previously known only for dimensions up to 6, as shown in Table 1.

n	s_n	c_n
1	1	2
2	2	4
3	4	6
4	7	8
5	13	14
6	26	26

Table 1: Maximum Lengths of Snakes and Coils

Above dimension 6, only upper and lower bounds have been reported in the literature. Table 2 shows several lower bounds for coils obtained by Abbott and Katchalski [Abbo91b], and Even [Even63]. Clearly, $c_n - 2 \le s_n$, since one can always form an open snake by deleting a node from a coil. The lower bounds for s_n in Table 2 for n = 7, 9, 10, and 11 follow from this. Three examples showing $88 \le s_8$ are given in [Abbo91b].

n	*l.b.* for s_n	*l.b.* for c_n
7	46	48
8	88	88
9	168	170
10	322	324
11	618	620

Table 2: Several Known Lower Bounds

A number of authors have derived upper bounds for c_n. The most recently reported are in [Snev93].

GA EXPERIMENT SETUP

Our genetic algorithm (GA) experiments take two different tacks based on the representation of a snake. In one, we simply use the hypercube coordinates (node numbers in decimal that correspond to the binary encoding) as the representation of individuals in the population. For example, from Figure 2 the node sequence of the coil is:

0 1 3 7 15 14 12 8

We set the length of an individual to a constant (e.g., to 60 for Q_7) and then create the initial population at random. The first node is randomly chosen from among all nodes. Subsequent nodes in the individual are selected at random from the appropriate nodes in the adjacency table (in Q_7 for example, we have a table with 128 entries representing each node and the corresponding 7 nodes adjacent to them). This insures that individuals in the initial population are valid paths. The fitness of each individual is based on the cube of the length of the longest subsnake in each.

Our other representation approach is based on the transition sequence [Klee67, Adel73] of a path. The transition sequence is related to the single bit position that changes from one node to the next adjacent node. For example, the transition value from node 3 (0000011) to node 7 (0000111) is 3, and the transition value from node 6 (0000110) to node 14 (0001110) is 4. The transition sequence for the coil in Figure 2 is:

1 2 3 4 1 2 3 4

Again, each individual in the initial population is created at random and is insured to be a valid path. Insuring valid paths is much easier with this approach since the need for the adjacency table (or an adjacency calculation) is eliminated. Also, the fitness is the cube of the longest subsnake in the sequence.

When using integer sequences that are order-based for individuals in the genetic algorithm, standard bit string genetic operators typically are inappropriate. Searching for snakes or traveling salesman routes demand that the operators that impact the order of the sequence maintain valid ordering. We use the enhanced edge recombination operator [Star91] with our node sequence representation because it focuses on maintaining node adjacency, an important feature for evolving long snakes. In addition, we use two variations of the evolutionary strategy. Namely, with the node sequence approach we use the standard mate selection and mating scheme where an entire next generation is created. We use the "one-at-a-time" replacement scheme used in *GENITOR* [Star91] for the transition sequence approach. In the one-at-a-time scheme, a new generation contains the same individuals as the previous generation except for the replacement of the worst (least fit) individual with the offspring of one mating pair of individuals selected in the usual fashion.

As for the specific GA parameter settings for the node sequence approach, we used a variety of population sizes ranging from 10,000 to 25,000, and a variety of crossover probabilities ranging from 0.6 to 0.95 with the enhanced edge recombination crossover scheme. We used a seeding scheme whereby we seeded the initial population of a trial with the best set of results from previous trials, although no fitness scaling was used. A trial ran for a specified number of generations and was then stopped. The limit here was in the neighborhood of 50,000 generations.

For the transition sequence approach, we used the same ranges for population size and crossover probabilities (i.e., 10,000 to 25,000, and 0.6 to 0.95). Mutation probabilities ranged from 0.01 to 0.04. We used a two-point crossover scheme with this approach. Again, seeding was used as with the node sequence approach. A trial was terminated whenever the population showed signs of convergence as indicated by no improvement in the best individual over several generations.

RESULTS

We use the Genetic Algorithm to search for long snakes (both open and closed) in Q_7 and Q_8 (we plan to continue our investigation in Q_9 and Q_{10} soon). Our heuristic approach is aimed at increasing basic knowledge of hypercubes by finding instances of open and closed snakes which exceed the current theoretical lower bounds. In addition, we hope to discover new lower bounds for s_9 and s_{10}, and new lower bounds for c_9 and c_{10}.

The best reported bounds known for c_7 are $48 \leq c_7 \leq 60$, as reported in [Deim85]†. This implies that $s_7 \geq 46$. The genetic algorithm was able to find an open snake of length 50 (see Figure 3). The companion exhaustive (enumerative) approach found all snakes in Q_7, the longest of which were of length 50. This established that $s_7 = 50$, and, in addition, established that either $c_7 = 48$ or $c_7 = 50$ [Koch94]. The GA was able to find a snake of this length using much less runtime than the enumerative algorithm. This is quite encouraging; the heuristic was able to find the optimal solution to a very hard problem; and was able to find it relatively quickly. This indicates that the genetic algorithm would be suitable for finding long snakes in higher dimensional hypercubes.

40 41 43 47 63 62 60 124 125 121 123
122 106 98 99 103 119 118 86 94 95 79
75 73 72 88 80 81 85 69 68 100 36 37
53 49 51 50 18 26 27 25 29 13 12 14 6
7 3 1 0

Figure 3. Snake of Length 50 in Q_7 Found by GA

† Although Solov'jeva's results [Solo87] for $n \geq 7$ are asymptotically better than Deimer's, Deimer's result for $n = 7$ is better.

32 0 8 24 28 30 31 23 21 53 49 57 41
45 13 77 93 89 81 65 97 101 103 119
115 123 107 75 11 3 35 163 179 147 211
195 199 197 213 245 241 249 233 201 137
153 157 189 191 255 223 222 220 216 208
192 224 228 230 246 242 250 234 202 138
130 134 150 148 180 176 184 168 172 174
46 38 54 50 18 82 66 70 68 84 116 112
120 104 108

Figure 4. Snake of Length 89 in Q_8 found by GA.

Applying the genetic algorithm in Q_8, we found an open snake that has length 89, implying that $s_8 \geq 89$ (see Figure 4). This improves on the current world record of 88 held by Abbott and Katchalski [Abbo91b]. Their search for long snakes is based on the construction of snakes and coils using information about the symmetry of the hypercube and known long snakes and coils from smaller dimensional hypercubes. This is in sharp contrast to our "blind" heuristic search technique.

Abbott and Katchalski also report the existence of a coil of length 88 in Q_8. In addition, they report on the existence of several long open snakes in Q_8 including one of length 86, two of length 87, and three having length 88. (Note, in their paper they count the nodes in an open snake rather than the edges as is generally done. For coils both counts yield identical results.)

Using the genetic algorithm, the Q_7 result and our new Q_8 result have come from an evaluation of fewer than two million paths (a very, very small fraction of the combined total search space), showing the strength and robustness of the genetic algorithm search when dealing with extremely difficult problems such as snake hunting. Exact computer evaluation of c_n or s_n for any $n \geq 8$ is, we feel, infeasible at this time, but the known lower bounds should be amenable to improvement by our heuristic approach. All we need are particular examples of long snakes, and for this the genetic algorithm approach can be very effective.

CONCLUSIONS

We are currently investigating improvements to our transition sequence based representation version because we feel that it will eventually lead to record setting results in dimension 9 and 10. Due to its simplicity, we expect to be able to improve the performance of the GA by introducing an additional GA operator such as Engineered Conditioning (EC) [Pott92a]. This results in what is known as a hybrid genetic algorithm because of the addition of the EC local improvement operator. We have achieved extensive improvements using EC when applied in the area of multiple fault diagnosis. In addition, we are investigating the use of other heuristic search schemes to find snakes and coils.

Our work on finding snakes in hypercubes was initially motivated by our analysis of various local improvement operators (e.g., EC, E^2C, H^kC*) discussed in [Mill93], and applied to multiple fault diagnosis. Our current results for snakes give the worst-case number of iterations for H^1C* which iteratively moves to the best immediate neighbor. H^2C* works in a similar fashion except that the neighborhood includes immediate neighbors as well as those two steps away (at a Hamming distance ≤ 2). We are modifying our hunt for these types of snakes, that is, those corresponding to H^kC for k = 2, 3, 4.

REFERENCES

[Abbo88]
Abbott, H.L., and M. Katchalski, "On the Snake-in-the-Box Problem", *J. Combin. Theory*, Vol. 45, pp. 13-24, 1988.

[Abbo91a]
Abbott, H.L., and M. Katchalski, "Further Results on Snakes in Powers of Complete Graphs", *Discrete Mathematics*, Vol. 91, pp. 111-120, 1991.

[Abbo91b]
Abbott, H.L., and M. Katchalski, "On The Construction Of Snake In The Box Codes", *Utilitas Mathematica*, Vol. 40, pp. 97-116, 1991.

[Adel73]
Adelson, L.E., R. Alter, and T.B. Curtz, "Long snakes and a characterization of maximal snakes on the d-cube", *Proceedings, 4th SouthEastern Conference on Combinatorics, Graph Theory and Computing*, Congr. Numer. 8, pp. 111-124, 1973.

[Danz67]
Danzer, L., and V. Klee, "Lengths of Snakes in Boxes", *J. Combinatorial Theory*, Vol. 2, pp. 258-265, 1967.

[Davi65]
Davies, D.W., "Longest -Separated- Paths and Loops in an N Cube," *IEEE Trans. Electronic Computers*, Vol. 14, p. 261, 1965.

[Davi91]
Davis, L., (ed.) *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York 1991.

[Deim85]
Deimer, K., "A New Upper Bound for the Length of Snakes," *Combinatorica*, Vol. 5(2), pp. 109-120, 1985.

[Doug69]
Douglas, R.J., "Upper Bounds on the lengths of cir-

cuits of even spread in the d-cube," *J. Combin. Theory*, Vol. 7, pp. 206-214, 1969.

[Even63]
Even, S., "Snake in the Box Codes," correspondence in *IRE Transactions on Electronic Computers*, Vol. EC-12, p. 18, 1963.

[Gold89]
Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Co., 1989.

[Hara88]
Harary, F., J. P. Hayes and H. J. Wu, "A survey of the theory of hypercube graphs", *Comput. Math. Applic.*, 15 (1988) 277-289.

[Holl75]
Holland, J.H., *Adaptation in Natural and Artificial Systems*, Ann Arbor: The University of Michigan Press, 1975.

[Kaut58]
Kautz, W.H., "Unit-Distance Error-Checking Codes," *IRE Trans. Electronic Computers*, Vol. 7, pp. 179-180, 1958.

[Klee67]
Klee, V., "A method for constructing circuit codes," *J. Assoc. Comput. Mach.*, Vol. 14, pp. 520-538, 1967.

[Klee70]
Klee, V., "What is the maximum length of a d-dimensional snake?" *Amer. Math. Monthly*, Vol. 77, pp. 63-65, 1970.

[Koch94]
Kochut, K.J., J.A. Miller, W.D. Potter, R.W. Robinson, "Hunting For Snake-In-The-Box Codes," submitted to *Annals of Mathematics and Artificial Intelligence*, 1994.

[Liep90]
Liepins, G.E., M.R. Hilliard, J. Richardson, and M. Palmer, "Genetic Algorithm Applications to Set Covering and Traveling Salesman Problems," in *OR/AI: The Integration of Problem Solving Strategies*, (Brown, ed.), 1990.

[Mill93]
Miller, J.A., W.D. Potter, R.V. Gandham, and C.N. Lapena, "An Evaluation of Local Improvement Operators for Genetic Algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 5, pp. 1340-1351, Sept/Oct 1993.

[Pott90a]
Potter, W.D., B.E. Tonn, M.R. Hilliard, G.E. Liepins, S.L. Purucker, and R.T. Goeltz, "Diagnosis, Parsimony, and Genetic Algorithms," in *Proceedings of the Third Int. Conf. Industrial & Engineering Applications of AI and Expert Systems*, pp. 1-8, July, 1990.

[Pott90b]
Potter, W.D., J.A. Miller, and O.R. Weyrich, "A Comparison of Methods for Diagnostic Decision Making," in *Expert Systems with Applications: An International Journal*, Vol. 1, pp. 425-436, 1990.

[Pott92a]
Potter, W.D., J.A. Miller, B.E. Tonn, R.V. Gandham, and C.N. Lapena, "Improving The Reliability of Heuristic Multiple Fault Diagnosis Via The EC-Based Genetic Algorithm," in *APPLIED INTELLIGENCE: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*, Vol. 2, pp. 5-23, 1992.

[Pott92b]
Potter, W.D., R. Pitts, P. Gillis, J. Young, and J. Caramadre, "IDA-NET: An Intelligent Decision Aid For Battlefield Communications Network Configuration," in the *Proceedings of the Eighth IEEE Conf. on Artificial Intelligence for Applications (CAIA'92)*, pp. 247-253, March 6, 1992.

[Sing66]
Singleton, R.C., "Generalized Snake-in-the-Box Codes," *IEEE Trans. Electronic Computers*, Vol. 15, pp. 596-602, 1966.

[Snev93]
Snevily, H.S., "The Snake-in-the-Box Problem: A New Upper Bound," *Discrete Math* (to appear).

[Solo87]
Solov'jeva, F.I., "An Upper Bound for the Length of a Cycle in an n-Dimensional Unit Cube", *Discret. Analiz.*, Vol. 45, 1987.

[Star91]
Starkweather, T., S. McDaniel, K. Mathias, D. Whitley, and C. Whitley, "A Comparison of Genetic Sequencing Operators," in *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 69-76, 1991.

[Tove81]
Tovey, C.A., "Polynomial Local Improvement Algorithms in Combinatorial Optimization," Ph.D. Dissertation, Stanford University, Palo Alto, CA, 1981.

[Wojc89]
Wojciechowski, J., "A new lower bound for Snake-in-the-Box codes," *Combinatorica*, Vol. 9, pp. 91-99, 1989.