ECAI 2012
Luc De Raedt et al. (Eds.)

doi:10.3233/978-1-61499-098-7-462

A New Approach to the Snake-In-The-Box Problem

David Kinny[1]

Abstract. The "Snake-In-The-Box" problem, first described more than 50 years ago, is a hard combinatorial search problem whose solutions have many practical applications. Until recently, techniques based on Evolutionary Computation have been considered the state-of-the-art for solving this deterministic maximization problem, and held most significant records. This paper reviews the problem and prior solution techniques, then presents a new technique, based on Monte-Carlo Tree Search, which finds significantly better solutions than prior techniques, is considerably faster, and requires no tuning.

1 INTRODUCTION

The "Snake-In-The-Box" (SIB) problem, first described more than 50 years ago [14], is a challenging combinatorial search problem with many useful applications. It consists in finding longest induced (chord-free) paths in the n-dimensional hypercube graph Q_n or equivalently, longest *codes* – sequences of length n bit vectors – that possess isomorphic adjacency properties. Open paths are usually called *snakes*, while closed paths (cycles) are called *coils* or *circuit codes*. We adopt these terms but alert the reader that in some areas of the literature, such as coding theory, the term snake used unqualified means a closed path, and open paths are called open snakes.

The n-dimensional hypercube graph Q_n is a highly symmetric undirected simple graph on 2^n nodes, each of which is a distinct *n-word*, an unsigned n-bit number in $[0,\ 2^n - 1]$. Each node is connected to n other adjacent ones, namely those that differ in exactly 1 bit, i.e. are at a Hamming distance of 1. A snake or coil is a certain type of self-avoiding path in the hypercube graph defined as follows.

Definition 1 ***(n-snake, n-coil)*** An open path (resp., cycle) in the n-dimensional hypercube Q_n is an *n-snake* (resp., *n-coil*) if every edge in Q_n between two nodes on the path is also an edge of the path. $\diamond$

Thus non-consecutive nodes in a snake or coil have a Hamming distance of 2 or more, and the *node sequence* of the path forms a type of Gray code. If the $n \cdot 2^{n-1}$ edges of Q_n are labelled (non-uniquely) with the index of the bit that changes in moving from one node to an adjacent one, the sequence of edge labels along a path is the path's *transition sequence*. We more formally define these complementary representations of snakes and coils in section 2.

A snake is *maximal* if it cannot be extended at either end, but maximal snakes vary in length for $n > 3$. The general SIB problem for dimension n is to establish the values of s_n and c_n, the lengths of longest n-snakes and n-coils respectively, and to exhibit suitable specimens. Longest snakes and coils identify longest sequences of n-words whose properties make them useful in applications including error-detecting codes, analog-to-digital encoders and converters, fault diagnosis in multiprocessor networks, modulation schemes in multi-level flash memories [27], and, somewhat remarkably, identifying gene regulation networks in embryonic development [28].

[1] Kyoto University, Japan. Email: dnk ∂ i.kyoto-u.ac.jp

Research on the SIB problem initially focused on applications in coding [14]. Coils are *spread-k circuit codes* for $k = 2$, in which n-words k or more positions apart in the code differ in at least k bit positions [12]. (The well-known Gray codes are spread-1 circuit codes.) Longest snakes and coils provide the maximum number of code words for a given word size (i.e., hypercube dimension).

A related application is in encoding schemes for analogue-to-digital converters including shaft (rotation) encoders. Longest SIB codes provide greatest resolution, and single bit errors are either recognised as such or map to an adjacent codeword causing an off-by-one error. So called *single-track* circuit codes [12] of spread 2, a special case of coils, are of particular interest as they also minimize the mechanical and electrical complexity of such encoders.

Finding optimal (longest) coils and snakes is hard because no general constructive method is known[2], so search-based techniques must be employed. The size of the search space is $O(n^{2^n})$; finding solutions by enumeration is trivial for $n \leq 6$, and was achieved for $n = 7$ more than 15 years ago [15, 19], but has not yet been done for $n = 8$ and is infeasible for larger n. Progress towards solving the SIB problem consists in finding better bounds on s_n and c_n, with empirical techniques contributing improved lower bounds, and analytical techniques contributing improved lower or upper bounds, which currently are not tight. Empirical research has focused primarily on finding safe search tree pruning heuristics, and on applying incomplete search techniques, principally Genetic Algorithms (GAs) and hybrid EC-based approaches. Section 3 contains a detailed review of previous approaches to the problem and their achievements.

Population based algorithms such as GAs and EC hybrids have been regarded as the state-of-the-art for the problem, but they require careful selection of representations and operators, and often need extensive tuning and have problems with scaling up. Other than the new bound of 98 for s_8 established in 2010, no improved bounds have been reported for such techniques since landmark results in 2005 and 2007. Developing better search tree pruning heuristics requires careful analysis, design and experimentation, and new constructive techniques require an advanced level of mathematical sophistication.

The main motivation for the research reported here was to determine how well minimally informed search techniques requiring little human design effort or tuning could perform on the SIB problem. To this end we adapted and applied a state-of-the-art stochastic tree search technique – Nested Monte-Carlo Search [6] – and were able to obtain significantly improved lower bounds for s_{10}, s_{11}, and s_{12} and better than order-of-magnitude speed-ups relative to prior EC-based approaches, as well as improved bounds for a special type of coils. Section 4 describes this technique and the results it has achieved, and is followed by conclusions and an appendix containing the transition sequences of the record-breaking snakes and coils.

[2] Known constructive methods produce long, but not necessarily optimal, snakes and coils from smaller ones, and typically require some search.

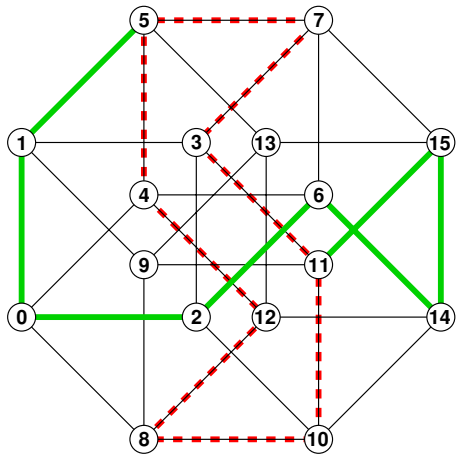

Figure 1. An optimal snake and coil in a dimension 4 hypercube.

2 SIB PROBLEM REPRESENTATION

A snake or coil is a (graph-theoretic) path, and can be represented either by its node sequence, or by its start node and transition sequence. Here we formalize these notions and others used in the sequel.

2.1 Representation of paths

Definition 2 ***(vicinity)*** Let $H(,)$ be the Hamming distance metric. The *vicinity* $V(a)$ of a node $a \in Q_n$ is $\{b \in Q_n : H(a,b) = 1\}$. The *vicinity* $V(A)$ of a set A of nodes in Q_n is $\bigcup_{a \in A} V(a)$. $\diamond$

Definition 3 ***(node sequence, optimality)*** The *node sequence* of an n-snake S of length $j \geq 0$ is the sequence of n-words $a_0 \ldots a_j$ on the path from its start node or *head* a_0 to its end node or *tail* a_j. S is *maximal in* Q_n if this sequence is not a subsequence of that of any other n-snake, and *optimal in* Q_n if no other n-snake is longer. An n-coil C with node sequence $a_0 \ldots a_j,\ \ j \geq 1,\ a_j \in V(a_0)$ has length $j+1$. C is *optimal in* Q_n if no other n-coil is longer. $\diamond$

We note in passing that coils are always of even length, a coil of length j can always be converted to a snake of length $j-2$ by deleting any node, and that a node sequence representing an optimal snake or coil in Q_n also represents a non-optimal snake or coil in Q_m for any $m > n$. Figure 1 depicts an optimal snake and coil in dimension 4, of length 7 and 8 respectively[3]. Deleting any node of this coil produces a maximal but sub-optimal snake of length 6.

Definition 4 ***(transition sequence)*** The *transition sequence* of an n-snake with node sequence $a_0 \ldots\ a_j$ is $\lg(a_0 \otimes a_1) \ldots\ \lg(a_{j-1} \otimes a_j)$, where $\otimes$ denotes exclusive-or and lg denotes the binary logarithm. The transition sequence of an n-snake of length 0 is the empty sequence. The transition sequence of an n-coil with node sequence $a_0 \ldots\ a_j$ is $\lg(a_0 \otimes a_1) \ldots\ \lg(a_j \otimes a_0)$. $\diamond$

Transition sequences have the same lengths as the snakes and coils that they represent. Transitions range from 0 to $n-1$, so they are a slightly more compact representation than node sequences. Transition sequences are sometimes called *change sequences* and their elements *change numbers*. The transition sequence of a snake or coil is invariant under translation, so standing alone it represents the equivalence class of 2^n snakes or coils that start from any node in Q_n.

For example, starting at node 8 and proceeding anticlockwise, the node sequence of the coil in figure 1 is $(8, 10, 11, 3, 7, 5, 4, 12)$, and its transition sequence is $(1, 0, 3, 2, 1, 0, 3, 2)$, which we shall write compactly as 10321032. This is an example of a *doubled* coil, one of length $2k$ whose transitions k apart are identical (also called a *symmetric* or *natural* coil). The 15 other doubled coils that share the same transition sequence start from each of the 15 other nodes.

Definition 5 ***(compatibility)*** A node $b \in Q_n$ is *compatible* with an n-snake with node sequence $a_0 \ldots a_j$ iff $b \notin V(\{a_i : 0 < i < j\})$. $\diamond$

Compatible nodes are those not in the vicinity of any node in a snake except its ends. A non-maximal snake may be extended by adding to it any compatible neighbour of either end. Adding a node in the vicinity of both turns the snake into a coil, which is necessarily maximal and cannot be extended. Note however that some papers dealing only with coils use maximal to mean optimal as we have defined it.

2.2 Canonical Representations

The hypercube Q_n is highly symmetric, possessing $n! \cdot 2^n$ automorphisms generated by the $n!$ possible permutations of its n axes and the 2^n possible translations of its origin. More precisely, $\mathrm{Aut}(Q_n)$ is the wreath product of the symmetric groups S_2 and S_n [10]. These symmetries allow any type of search process for snakes and coils to reduce the size of the search space by restricting attention to those whose transition sequences are in some kind of some canonical form.

A first step is to define snakes and coils with identical transition sequences as equivalent, taking those that start at node 0 as the representatives of their class. Thus the coil $(8, 10, 11, 3, 7, 5, 4, 12)$ in figure 1 would be represented by the coil $(0, 2, 3, 11, 15, 13, 12, 4)$. Defining a canonical representation that is also invariant under permutation of hypercube axes (or equivalently, permutation of bit positions in n-words) is usually done by choosing from each equivalence class induced by such permutations the lexically smallest transition sequence, or equivalently, requiring that each transition number first appear in a transition sequence only after all smaller ones have appeared. Every transition sequence can be canonicalized by applying an appropriate permutation. For example, the canonical form of the coil transition sequence 10321032 is 01230123, obtained by applying the permutation (01)(23). A canonical transition sequence in dimensions 4 and above will always begin with 0120 or 0123. The prefixes 0121 and 0122 are not valid chord-free paths, and 0124, 0125, etc., are not lexically smallest.

As mentioned, coils are always of even length because each transition number must occur an even number of times in their transition sequences in order to form a cycle. A coil of length j also has j *rotational variants*, obtained by rotation of its node sequence. One can define more restricted canonical forms for coils by considering rotational variants to be equivalent and employing a lexicographic ordering to distinguish one of the variants as the canonical form [13].

3 PREVIOUS APPROACHES AND RESULTS

Since a snake or coil is just a sequence of numbers satisfying specific adjacency and separation constraints, many solution techniques are applicable. One can search using complete search algorithms such as depth-first search (DFS) with pruning and move ordering heuristics, or encode the constraints as propositional Boolean formulae and use a SAT solver or related techniques such as Answer Set Programming.

[3] To be precise, figure 1 could be said to depict 2 snakes and 16 coils, since the start nodes and the direction of traversal of the cycle are not marked. As snakes and coils are paths, they are technically distinct from their inverses and, in the case of coils, from their rotational variants.

Table 1. Proven values of s_n, c_n and d_n for $2 \leq n \leq 7$

n	2	3	4	5	6	7
s_n	2	4	7	13	26	50
c_n	4	6	8	14	26	48
d_n	4	6	8	14	26	46

As the search space grows superexponentially as $O(n^{2^n})$, incomplete stochastic search techniques such as Genetic Algorithms and Monte-Carlo techniques become necessary when $n > 7$. Problem-specific techniques based on deeper mathematical insights have been developed which construct longer solutions from shorter ones already known or discovered by search. Many of the above approaches have also been used to find spread-k codes for $k > 2$ [13, 29]. In this section we review SIB solution techniques which have been reported. We commence with a review of known bounds.

3.1 Known bounds

Analytical lower and upper bounds for the length c_n of longest coils in Q_n are obtained by some form of counting argument, which also establishes a lower bound on the length of optimal snakes by $s_n \geq c_n - 2$. Discovery of long snakes and coils establishes empirical lower bounds, and full enumeration establishes values of c_n and s_n directly. Table 1 contains proven optimal values [8, 15, 19] for dimensions 2 to 7, including values d_n for the special case of doubled coils [2, 3]. The best general lower bound reported is

$$c_n \geq \frac{77}{256}2^n = 77 \cdot 2^{n-8} \approx \frac{3}{5}2^{n-1} \quad (n \geq 11)$$

established more than 20 years ago [1]. Analytical upper bounds for c_n have been incrementally refined [24, 23, 17] as described in [9], which provides the best known bound for $36 < n < 19080$

$$c_n \leq 2^{n-1}(1 - \frac{1}{4n-9}) \quad .$$

All known upper bounds approach 2^{n-1} as $n \to \infty$, so for $n > 3$ the spread is wide with $\frac{3}{5} \leq \frac{c_n}{2^{n-1}} \leq 1 - \epsilon$. In fact, the trend in c_n is decreasing away from 2^{n-1}, with $c_7 = 48 < \frac{4}{5}2^6$. It is an open question whether an asymptotic upper bound less than 2^{n-1} exists.

Table 2 contains empirical lower and analytical upper bounds for s_n, c_n and d_n in dimensions 8 to 15, where exact values are unknown. Earlier values in parentheses give an indication of progress. It is likely that the current lower bounds for dimension 8 are optimal. It is also clear that very little progress has been made in improving empirical lower bounds in the higher dimensions.

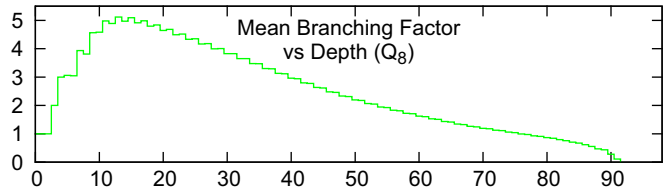

Figure 2. Mean branching factor vs depth for randomized search in Q_8.

3.2 Basic Complete Search Technique

Basic search starts with a short snake S such as (0), or $(0, 1, 3, 7)$ corresponding to the canonical transition sequence prefix 012, and repeatedly chooses a compatible neighbour v of the tail u to extend the path so that $S' = Sv$ is a canonical coil or snake with new tail v. Search terminates or backtracks when no such node v can be found or a coil is formed. Building a path in this way can be viewed as a finite single-player game whose goal is to achieve maximum length by choosing at each step from the available options (legal moves). The choice is between the at most $n-2$ and on average $\approx (n-2)/3$ nodes in $V(u)$ compatible with S. Each time a new tail is chosen, all other nodes in $V(u)$ become incompatible with the new snake S'.

The branching factor of the resulting search tree decreases with depth as more nodes become incompatible with the growing snake and the average number of options is reduced, as is seen in Figure 2. Searching only for canonical paths safely prunes some higher level branches. It should also be clear that such a search tree is transposition free, i.e., all (canonical) snakes occur exactly once in the tree, and that the maximum depth of the tree is exactly $\max(s_n, c_n)$. Search can easily be parallelized as subtrees are independent.

For fixed dimension n, nodes' neighbours can be precomputed and stored in an array of size $n \cdot 2^n$, or efficiently computed on-the-fly by bitwise operations. To determine compatible nodes it suffices to maintain an array of size 2^n holding for each node the number of its neighbours already in the snake, incrementing counts when a node is added and decrementing them when backtracking. While other tree-search algorithms are applicable, DFS has the well-known advantage that only memory linear in the search depth is required, and this too may be pre-allocated as the search depth is bounded by 2^{n-1}.

Memory needed by DFS for SIB is very modest, however DFS search times grow superexponentially in n. For example, on a single core of a 2.26GHz Xeon E5520 CPU, DFS finds the unique optimal 6-snake in milliseconds, finds an optimal 7-snake in 36 seconds, and finds an 8-snake of length 82 in 1 minute, but a further day of search only improves this to 86, still far from the optimal $s_8 \geq 98$.

Table 2. Previous and current analytical and empirical bounds on s_n, c_n and d_n for $8 \leq n \leq 15$

n	l.b. $77 \cdot 2^{n-8}$	Empirical lower bound on s_n	Emp. lower bound on c_n	Emp. lower bound on d_n	Analytical u.b. on c_n
8	-	$(88^a, 89^b, 94^c, 97^d)$ 98^g	(88^a) 96^c	$(86^\diamond, 90^\star)$ 94^h	123 †
9	-	$(168^a, 186^e, 188^f)$ 190^h	$(170^a, 180^e)$ 188^h	$(122^i, 156^\star, 170^h)$ $180^\star$	249 †
10	-	$(322^a, 338^c, 358^e, 363^f)$ $370^\star$	$(324^a, 340^c, 344^e)$ 348^h	$(238^\star, 294^h)$ 348^h	500 †
11	616	$(618^a, 680^e)$ $695^\star$	$(620^a, 630^e)$ 640^h	(494^h) 640^h	994 ‡
12	1232	$(1236^a, 1260^e)$ $1274^\star$	1238^a	(902^h) $1128^\star$	1995 ‡
13	2464	2466^a	2468^a	$1896^\star$	4000 ‡
14	4928	4932^a	4934^a	?	8017 ‡
15	9856	9866^a	9868^a	?	16062 ‡

◇ (Adelson et al. 1973), a (Abbott & Katchalski 1991), b (Potter et al. 1994), c (Paterson & Tuliani 1998), d (Rajan & Shende 1999), e (Casella & Potter 2005), f (Tuohy et al. 2007), g (Carlson & Hougen 2010), h (Wynn 2009–12), i (Potter (unpub.) 2011), ⋆ (this research), † (Solov´jeva 1987), ‡ (Lukito 1998)

3.3 Priming, Pruning and Move Ordering

Priming (or seeding) consists in choosing a longer snake as the starting point for search, so restricting search to a subtree of the entire tree. Long snakes in the next lower dimension are often good choices.

For example, of the 12 optimal canonical 7-snakes of length 50, one can be extended to a near-optimal 8-snake of length 97, which can be found by DFS in under a minute if the former is used as the primer. However, such a snake has a specific structure, with the highest transition number 7 first occurring past the middle of the path. It is impossible to obtain an 8-snake of length 98 by extending a 7-snake of length 50. This means the length of the prefix entirely in Q_7 of any 8-snake of length 98 is less than 50, perhaps considerably so. For example, the Q_7 prefix of the record length 98 snake reported in [4] has length 26, and its longest extension in Q_7 has length 47. To have selected this particular prefix as a search primer is highly unlikely, and priming is an inherently unsafe method for finding optimal snakes. Nonetheless, priming the search with near-optimal snakes in the next lower dimension is a widely-used technique to make search tractable.

As mentioned, more restrictive canonical forms for coils can be defined that consider all rotational variants to be equivalent and distinguish one as the canonical representation, allowing branches that lead to non-canonical forms to be safely pruned if searching only for coils. Testing for this can be somewhat costly; a number of algorithms to do so were recently described and evaluated [13]. Similarly, branches deep in the tree that cannot possibly be extended to form a coil (due to the lack of any feasible path back to 0) can safely be pruned [13, 15], as can those where the tail is too far from 0 to return within available moves (if searching for coils of a specific length).

Unsafe pruning heuristics based on statistical properties of observed snake transition sequences have also been devised. The record lower bound for s_{10} that held until this research was found by a manual hyper-heuristic approach that used DFS with unsafe pruning heuristics, based on "tightness" and its rate of change [25], obtained from an analysis of model snakes generated by EC techniques [5].

The well-known technique of *move ordering* can also be applied. This consists in sorting the possible moves at each step of the search process by some ranking criteria and choosing a best option initially, choosing a next best on backtracking, etc. Well-chosen ranking criteria can dramatically reduce the time to find good solutions, but it is problematic to find static criteria that apply uniformly at all depths in the search tree. We are not aware of any application of dynamic (e.g., depth-dependent) move ordering heuristics to the problem.

3.4 Meta-heuristic and SAT-based approaches

Potter pioneered the application of a Genetic Algorithm (GA) to the problem, using crossover operators on large populations to find a length 50 snake in Q_7 whose optimality was later established by enumeration, and to improve the lower bound on s_8 to 89 [19]. His research group subsequently established improved bounds for s_9 to s_{12} and c_9 to c_{11} using a "Population Based Stochastic Hill Climber" [5], and by DFS with the unsafe pruning heuristic technique described above [25]. These results have only recently been surpassed by this research and that of Wynn [26]. The current record lower bound of 98 for s_8 was also obtained by GA techniques [4]. Other EC techniques such as Ant-Colony algorithms and Particle Swarm Optimization have also been applied. These EC techniques have been effective but have the disadvantage that they may need extensive trial-and-error tuning [25]. Automated hyper-heuristic approaches are clearly applicable here, but to our knowledge have not yet been applied.

Encoding adjacency constraints as CNF Boolean formulae and applying modern propositional SAT solvers is another search technique which has been successfully applied to the more general spread-k code problem, finding many improved bounds for codes with $k > 2$ but no new snakes or coils [7, 29]. Search is for codes of a specific length fixed by the encoding. Disadvantages are that the number of variables ($O(n \cdot 2^n)$) and constraints is large and scaling up is poor: an attempt to repeat Kochut's enumeration of optimal 7-coils [15], recently done in 2.4 days [26], did not complete in 100 days [29].

3.5 Constructions based on Symmetries

Techniques based on theoretical insights have been developed which more efficiently construct snakes and coils with certain symmetries. Recursive constructions include [1]. A non-recursive method based on *necklaces*, classes of codes equivalent under cyclic rotation, was used to find many new spread-k codes including record breaking coils (and hence snakes) in Q_8 and Q_{10} [18]. The length 96 8-coil was derived from 8 permuted and/or inverted repetitions of an initial subsequence of length 11, suitably joined. The technique was later generalised and constructions based on linear codes developed [11].

The most significant recent progess in constructive techniques is reported by Wynn [26], who found record coils in Q_9 to Q_{11}, record doubled coils in Q_8 to Q_{12}, and a record snake in Q_9. Coils were formed as permuted repetitions of an initial transition sequence found by search: the highly symmetrical length 640 11-coil consists of 20 such length 32 subsequences. The technique first searches for combinations of target nodes and permutations that fix a feasible *coil skeleton*, then for a subsequence fitting the skeleton. If the initial subsequence is repeated just once, permuted by the identity permutation, the result is a doubled coil. A technique for constructing an $(n+1)$-snake from two n-snakes by incrementally searching for a permutation that makes the second snake compatible with the first is also given, and was used to find the record 9-snake of length 190.

4 A NEW STOCHASTIC SEARCH TECHNIQUE

The basis of our approach is Monte-Carlo Tree Search (MCTS). MCTS techniques, especially UCT [16], have led to major advances in computer play in difficult two-player games such as Go and Amazons, and have more recently been applied successfully to single-player games. One variant, Nested Monte-Carlo Search (NMCS), has recently broken long-standing records for hard single-player games such as Morpion Solitaire and crossword-puzzle construction [6], and been applied to other hard combinatorial search problems [21].

4.1 Nested Monte-Carlo Search

NMCS is a recursive stochastic k-level tree-search algorithm. A *playout* in NMCS is a path that descends the search tree from an initial state to a leaf node by making a random choice of move at each level. The choice may be uniform, or biased in some way. Playouts may be performed millions of times a second. NMCS operates as follows.

- Search starts at level $k > 0$, which calls level $k-1$ to evaluate the states that result from each move possible in a given state, then chooses a move with a highest valuation. It does this repeatedly until it reaches a leaf node, making bd calls to the level below, where b is the average branching factor and d the average depth of a leaf node. It returns the path found and its value to its caller.
- The base level (level 0) estimates the value of a state by performing a single playout, returning the path found and its value.

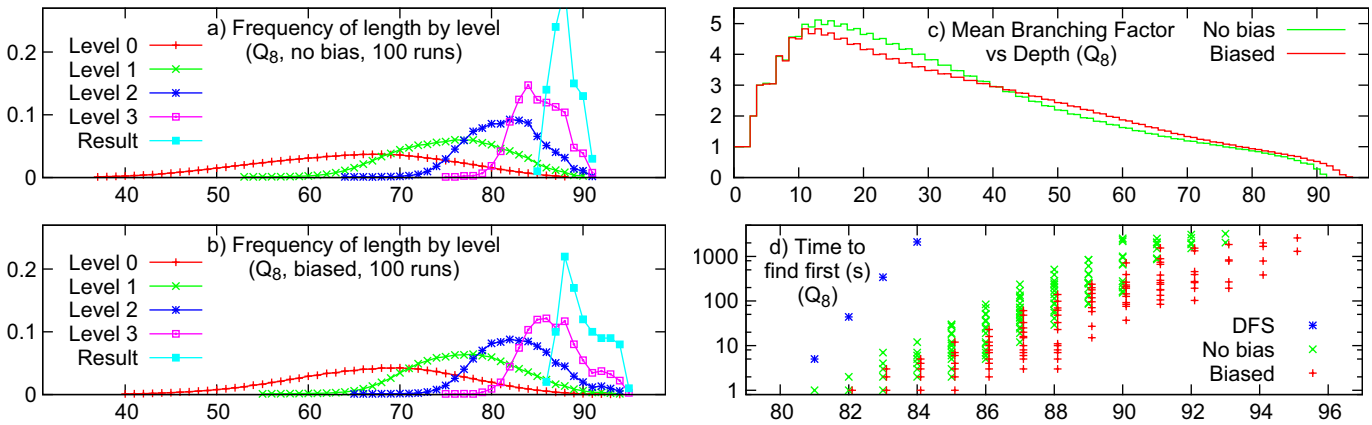

Figure 3. Effect of NMCS playout bias on: a,b) path length distribution; c) branching factor; d) time to find a snake of a given length (seconds).

- Level $k > 0$ memorizes the best valued path returned by level $k-1$, replacing it whenever an improvement is found. It uses this memorized path to choose a move if the path's value is better than those returned by subsequent calls to level $k-1$, ensuring that it always chooses the best solution found so far.

The total number of level 0 playouts performed is $O((bd)^k)$. Details of the NMCS algorithm and a comparison with a related adaptive MCTS algorithm may be found in [6] and [22]. NMCS effectively has no tunable parameters, as choosing the number of levels k is equivalent to specifying an approximate search timeout.

4.2 NMCS for the SIB problem

A straighforward adaptation of unbiased NMCS to the SIB problem was implemented, and proved capable of finding optimal 7-snakes in under 2 minutes with no primer. As expected of a stochastic algorithm, the time taken to find a snake and the specific snake found varied from one run to the next, as the pseudo-random number source used was seeded from the system clock.

Run on a single core of a 2.26GHz Xeon E5520 CPU, near-optimal 8-snakes of length 97 were found in less than a day with no primer, and in under 10 minutes using near optimal 7-snakes as primers. 9-snakes of length 188 were found in a few hours using near optimal 8-snakes as primers, but results from unprimed and higher dimensional searches were disappointing. It was clear that judicious priming was a key element of scaling up to higher dimensions.

Playout Policy. In MCTS, a move ordering heuristic which biases the level 0 playout with problem-specific knowledge is known as a playout policy. Effective exploration of the search space requires sufficient randomness in the policy, and feature-weighted policies such as Boltzmann softmax have proven effective [21]. For the SIB problem, a useful feature of a state is the number of nodes still compatible with the growing snake, since the larger this is the more likely it is in general that a relatively longer path can be found from the state.

However, in any given state all options are equivalent with respect to this measure, since the same set of compatible neighbours of the tail become incompatible no matter which of them is chosen to be the new tail. The needed measure is based on 1 level of lookahead: the number of options in each possible successor state, i.e., its branching factor in the game tree. Choosing a move that minimizes this measure minimizes the number of nodes eliminated at the subsequent step, except that if the measure is 0, the successor state is a dead-end, to be avoided unless no better option is available.

In the absence of any other clearly useful features, a very simple playout policy was implemented. If more than one option exists, then with probability p, one is chosen uniformly at random, and with probability $1-p$, one leading to a state with minimal (but non-zero) successors is chosen, breaking any ties uniformly at random. An efficient implementation of this policy required an additional array of size 2^n holding for each node how many of its neighbours are still compatible with the growing snake. For each node that becomes incompatible, the count is decremented for all its neighbours.

Effects of Bias. Using this policy with $p = 0.5$ led to significant improvement over unbiased playouts, reducing the time to find solutions of a given length and so increasing the length of solutions found for a given number of levels. While like-vs-like comparisons are not possible, the previous record 10-snake (363) took 2 weeks search to find [25]. Biased 6-level NMCS found length 363 10-snakes in 6.3 hours, about 50 times faster, extended the record to 367 in under 2 days, and eventually found a length 370 snake in 2 weeks of search. Both these techniques used near optimal 9-snakes as primers.

Search times for previous record length 11- and 12-snakes (680, 1260) were reported as weeks to months [5]. Using $(n-1)$-snakes as primers, biased 5-level NMCS found snakes of the the same lengths in under a day, a substantial speedup, and new record length 11-snakes (695) and 12-snakes (1274) in 4.1 and 3.2 days respectively. Transition sequences of these new record snakes are in the appendix.

The effects of bias can be seen in figure 3. Figures 3a and 3b show the distribution of path lengths found by unprimed 4-level NMCS in Q_8 for the unbiased and biased cases. The distributions are averaged over 100 runs, each of which took 2 minutes and perfomed about 60M playouts. Figure 3c shows the effect of bias on branching factor averaged over 2.6B playouts. Bias reduces the branching factor at low to medium depth, while increasing it at greater depth and increasing the maximum depth found. Figure 3d shows the most significant effect, a reduction of up to an order of magnitude in the time to find the first snake of a given length, as compared to unbiased playouts.

Doubled Coils. The NMCS-based algorithm was next adapted to search for doubled coils. Space constraints preclude an adequate exposition here, but initial results broke records for Q_8 to Q_{10}, including a result in Q_8 that had apparently stood for 38 years, however these were all shortly improved upon by Wynn's construction [26]. Subsequent refinement of the algorithm led to new record doubled coils for Q_9, Q_{12} and Q_{13} (see table 2). The 12-coil may be found in the appendix, and the other doubled coils found at `http://www.ai.soc.i.kyoto-u.ac.jp/~dnk/snakepit.html`.

5 CONCLUSIONS

NMCS, a stochastic tree-search algorithm, has been adapted to the Snake-In-The-Box problem, a challenging, scalable maximization problem, establishing new record lower bounds on the length of optimal snakes in dimensions 10, 11 and 12, and of doubled coils in dimensions 9, 12 and 13. This demonstrates that minimally informed search techniques that require no tuning can significantly outperform the EC-based techniques which had been regarded as state-of-the-art for the problem. Not only were improved bounds found, but the time to find snakes of a given length was more than an order of magnitude smaller than times previously reported.

ACKNOWLEDGEMENTS

This research was supported by the Ishida/Matsubara Laboratory of the Department of Social Informatics, Kyoto University.

A TRANSITION SEQ'S OF RECORD SNAKES

A snake of length 370 in Q_{10}:

```
01231 04132 51341 23104 31251 64152 10413 02140 15214 27014
52014 03240 14561 40152 10413 02140 15236 10364 31678 61302
34015 21041 30214 01523 61032 15201 40324 01453 76231 40163
01403 24016 51463 40230 14302 36213 04102 51498 41520 14031
26320 34103 20436 41561 04230 41036 10413 26723 41032 15201
40324 01462 14012 51043 24125 10412 58031 40163 01403 24016
51463 40230 14302 57235 43014 02340 14361 40230 15326 10230
42640 14752 13041 02510
```

A snake of length 695 in Q_{11}:

```
01234 12531 05215 34263 01340 25123 01251 34530 15671 20134
02512 30125 13452 60254 20310 25103 15435 20825 34251 30125
13451 25310 36243 51250 13521 54351 73543 10351 23013 54310
26352 03125 34513 01254 95210 52314 05130 12501 65125 01352
15435 21035 18045 32503 54310 32631 25345 13012 50130 24371
20132 43152 10316 35013 52305 34523 01326 08154 20310 52103
15435 21362 54310 35123 01354 31025 12397 8a658 24512 03125
10240 32512 63103 25421 53210 31543 02814 53215 30523 54156
31253 45130 12501 30243 71201 32431 52103 16350 13523 05345
23013 26081 54203 10521 03154 35213 62543 10351 23013 54310
25187 10591 53265 31045 01251 34530 15345 26530 13523 05345
26703 12534 51301 25013 06530 13543 10250 35431 03542 37821
52013 45310 32153 01345 26312 53451 30125 01302 45127 32152
01345 31032 15301 34516 23103 25432 03523 10235 43201
```

A snake of length 1274 in Q_{12}:

```
01230 45671 63123 08317 83287 58132 10360 18312 80186 34286
21698 74016 09289 73621 80168 31802 58674 81728 4a326 38213
25718 97832 80386 23169 12610 93809 26731 20647 86136 82163
14231 8623a 08a57 31052 80189 21328 16231 20162 31682 a0826
13952 18012 68167 12681 08213 87482 09286 32138 76932 81028
61378 13218 62803 40826 7a312 68316 78162 31280 18623 16438
01230 26321 08123 18731 23862 31280 18621 35013 26831 67816
20632 68108 26813 a8162 81386 18012 68167 12681 08213 26832
13659 06a78 13261 83708 65183 21361 83128 01862 17680 12302
63210 81231 620a2 80183 12801 86176 28012 86213 57628 13210
86712 68108 21326 7a623 16813 78326 82108 34631 26810 82138
16312 37632 10813 86321 08123 1620a 37280 12875 82109 28b26
71263 92862 90321 36183 12803 90826 31268 21082 67123 16806
31238 62312 80129 63081 62813 86180 34230 12386 23128 01861
76218 63083 26831 86a18 32130 41321 80168 31678 16231 28018
62316 27928 06312 38623 12801 86217 68012 31826 13218 01236
94632 10812 31628 13210 86712 68108 21309 86321 08123 16281
32108 27612 31680 16318 61230 a2863 21081 23162 81321 08671
26810 82132 68321 36095 14381 26834 8a367 16237 03268 21061
23610 82046 31268 10821 38163 12381 a6813 86230 73268 10821
32681 26082 6a162 31801 23182 61321 80123 14208 16231 6813a
02310 81683 18267 83123 86231 28018 62176 81928 16781 23083
26821 08132 81086 78132 18613 81086 7a312 68108 21326 83213
78132 18012 36831 80123 18061 a4628 63210 81231 62813 21086
71268 10821 32683 21306 826a1 62863 21081 23162 81321 08276
12316 80163 18612 36180 1632
```

A doubled coil of length 1128 in Q_{12}:

```
01231 43563 47105 18590 5325a 108b4 53812 9b314 694a0 3b859
28021 320a8 518b1 65129 1785b 4012b 89642 80360 7524b 169b5
98295 7b18b 48318 968ab 14058 94b63 09b52 35a19 08436 416b9
61a48 b40b2 8b63b 74805 1b604 32564 19217 865b0 23083 46387
0b405 49b43 24a0b 51843 50291 30869 8ab31 54925 b2032 ba540
51064 02907 5418b 02159 6825b 36b74 28106 91495 29471 05185
30596 5a108 b4598 163b9 14234 a09b8 53680 61960 a8518 b1251
63178 5b401 6b832 46809 20756 4b123 b5386 357b1 8b489 18328
ab140 5834b 2903b 5695a 13048 92412 b321a 48b40 b68b2 9b748
051b2 04965 24136 17825 0b690 89429 870b4 0543b 4964a 0b518
49506 31908 238ab 91543 65b60 96ba5 40510 24063 07541 8b061
53286 5b92b 7468  (twice)
```

REFERENCES

[1] H.L. Abbott and M. Katchalski, 'On the construction of snake in the box codes', *Utilitas Mathematica*, **40**, 97–116, (1991).

[2] L.E. Adelson, R. Alter, and T.B. Curtz, 'Computation of d-dimensional snakes', in *Procs. 4th S-E Conf. Combinatorics, Graph Theory and Computing*, pp. 135–139, (1973).

[3] L.E. Adelson, R. Alter, and T.B. Curtz, 'Long snakes and a characterisation of maximal snakes on the d-cube', in *Procs. 4th S-E Conf. Combinatorics, Graph Theory and Computing*, pp. 114–124, (1973).

[4] B.P. Carlson and D.F. Hougen, 'Phenotype feedback genetic algorithm operators for heuristic encoding of snakes within hypercubes', in *Proceedings of GECCO'10, Portland, Oregon*, pp. 791–798. ACM, (2010).

[5] D. Casella and W. Potter, 'New lower bounds for the snake-in-the-box problem: Using evolutionary techniques to hunt for snakes and coils', in *Proceedings of the Florida AI research Society Conference*, (2005).

[6] Tristan Cazenave, 'Nested monte-carlo search', in *Proceedings of IJCAI'09*, pp. 456–461, (2009).

[7] Yury Chebiryak and Daniel Kroening, 'An efficient SAT encoding of circuit codes', in *Procs. IEEE International Symposium on Information Theory and its Applications, New Zealand*, pp. 1235–1238, (2008).

[8] D. Davies, 'Longest 'separated' paths and loops in an n cube', *IEEE Transactions on Electronic Computers*, **14**(261), (1965).

[9] P.G. Emelyanov and A. Lukito, 'On the maximal length of a snake in hypercubes of small dimension', *Discrete Math.*, **218**, 51–59, (2000).

[10] F. Harary, *Graph Theory, 3rd edn*, Addison Wesley, 1972.

[11] Loeky Haryanto, *Constructing Snake-in-the-box Codes and Families of such Codes Covering the Hypercube*, Ph.D. dissertation, Delft University of Technology, 2007.

[12] Alain P. Hiltgen and Kenneth G. Paterson, 'Single-track circuit codes', *IEEE Transactions on Information Theory*, **47**, 2587–2595, (2000).

[13] Simon Hood, Daniel Recoskie, Joe Sawada, and Chi-Him Wong, 'Snakes, coils, and single-track circuit codes with spread k', 2011. Available at http://www.socs.uoguelph.ca/˜sawada/papers/coil.pdf

[14] W. H. Kautz, 'Unit-distance error-checking codes', *IRE Transactions on Electronic Computers*, **7**(2), 179–180, (June 1958).

[15] K. J. Kochut, 'Snake-in-the-box codes for dimension 7', *Combinatorial Mathematics and Combinatorial Computations*, **20**, 175–185, (1996).

[16] L. Kocsis and C. Szepesvàri, 'Bandit based monte-carlo planning', in *Proceedings of ECML'06, LNCS vol. 4212*, pp. 282–293, (2006).

[17] Agung Lukito, 'Upper and lower bounds for the length of snake-in-the-box codes', TR 98-07, Delft University of Technology, (1998).

[18] K.G. Paterson and J. Tuliani, 'Some new circuit codes', *IEEE Transactions on Information Theory*, **44**(3), 1305–1309, (May 1998).

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

[20] D.S. Rajan and A.M. Shende, 'Maximal and reversible snakes in hypercubes', *Proceedings of the 24th Australasian Conference on Combinatorial Mathematics and Combinatorial Computation*, (1999).

[21] Arpad Rimmel, Fabien Teytaud, and Tristan Cazenave, 'Optimization of the nested monte-carlo algorithm on the traveling salesman problem with time windows', in *Applications of Evolutionary Computation - Proceedings of Evo* 2010, Springer LNCS Vol. 6025*, (2010).

[22] Christopher D. Rosin, 'Nested rollout policy adaptation for monte carlo tree search', in *Proceedings of IJCAI'11*, (2011).

[23] Hunter S. Snevily, 'The snake-in-the-box problem: A new upper bound', *Discrete Mathematics*, **133**(1-3), 307–314, (1994).

[24] F.O. Solov´jeva, 'An upper bound for the length of a cycle in an n-dimensional unit cube', *Diskret. Anal.*, **45**, 71–76, (1987).

[25] Daniel R. Tuohy, Walter D. Potter, and Darren A. Casella, 'Searching for snake-in-the-box codes with evolved pruning models', in *Procs. 2007 Conf. on Genetic and Evolutionary Methods*, pp. 25–29, (2007).

[26] Ed Wynn, 'Constructing circuit codes by permuting initial sequences', 2012. Available at http://arxiv.org/pdf/1201.1647

[27] Yonatan Yehezkeally and Moshe Schwartz, 'Snake-in-the-box codes for rank modulation', 2011. Available at http://arxiv.org/abs/1107.3372

[28] Igor Zinovik, Yury Chebiryak, and Daniel Kroening, 'Periodic orbits and equilibria in glass models for gene regulatory networks', *IEEE Transactions on Information Theory*, **56**(2), 805–820, (Feb 2010).

[29] Igor Zinovik, Daniel Kroening, and Yury Chebiryak, 'Computing binary combinatorial gray codes via exhaustive search with sat solvers', *IEEE Transactions on Information Theory*, **54**(4), 1819–1823, (2008).