Improved Branch-and-Bound for
Low Autocorrelation Binary Sequences

S. D. Prestwich

Abstract The Low Autocorrelation Binary Sequence problem has applications in telecommunications, is of theoretical interest to physicists, and has inspired work by many optimisation researchers because of its difficulty. For many years it was considered unsuitable for solution by metaheuristics because of its search space topology, but in recent years metaheuristics have found long high-quality sequences. However, complete search has not progressed since a parallel branch-and-bound method of 1996. In this paper we find four ways of improving branch-and-bound, leading to a tighter relaxation, faster convergence to optimality and better scalability. We also extend known optimality results for skew-symmetric sequences from length 73 to 89.

Keywords autocorrelation · binary sequence · branch-and-bound · skew-symmetry
1 The LABS problem

Consider a binary sequence $S = (s_1, \ldots, s_N)$ where each $s_i \in \{1, -1\}$. The off-peak autocorrelations of $S$ are defined as

$$C_k(S) = \sum_{i=1}^{N-k} s_i s_{i+k} \quad (k = 1 \ldots N - 1)$$

(1)

and the energy of $S$ as

$$E(S) = \sum_{k=1}^{N-1} C_k^2(S)$$

(2)

The low-autocorrelation binary sequence (LABS) problem is to assign values to the $s_i$ such that $E(S)$ is minimized. A common measure of sequence quality is the merit factor $F(S) = N^2 / 2E(S)$. Theoretical considerations [Golay 1982] give an upper bound on $F(S)$ of approximately 12.32 as $N \to \infty$, and empirical curve fitting on known optimal sequences [Mertens 1996] yields an estimate of $F \approx 9.3$ for large $N$. This problem has many practical applications in communications engineering, and is of theoretical interest to physicists because it models 1-dimensional systems of Ising-spins. LABS has also generated interest among researchers from other fields who are interested in hard optimisation problems, and is problem number 5 in the CSPLib benchmark library [Gent and Walsh 1999], a web-based collection of constraint problems (though it has no constraints). Analytical methods have been used to construct optimal sequences for certain values of $N$ (see [Mertens and Bessenrodt 1998] for example) but for the general case search is necessary. Two possibilities are complete and incomplete search.

Complete search usually involves the enumeration of possibilities by backtracking. [Golay 1982] used exhaustive enumeration to find optimal sequences for $N \leq 32$. [Reinholz 1993] used a complete enumeration method with an accelerated function evaluation to compute exact solutions for $N \leq 39$. [Mertens 1996] enumerated optimal sequences for $N \leq 48$ using complete search augmented with two techniques to reduce the size of the search tree: branch-and-bound and symmetry breaking. Symmetry breaking, sometimes called symmetry exclusion, exploits the fact that sequences occur in equivalence classes of size 8 (see Section 2.6). However, even with these enhancements, complete search is unlikely to scale up to large sequences, and it is conjectured that for $N > 100$ progress will be made through mathematical insight rather than computer power [Mertens 1996]. The only other complete approach we know of is a recent Quadratic Programming model [Kratica 2012] which turned out to be much slower than Mertens’ method. It was only used to solve instances up to $N = 30$ by which time it took tens of thousands of seconds, whereas Mertens’ method took a few seconds on an older machine.

When complete search becomes impractical it is common to resort to incomplete methods such as simulated annealing, evolutionary algorithms, neural networks, ant colonies or greedy algorithms, which are often able to solve much larger instances. Unfortunately they perform quite poorly on some problems, and finding optimal LABS solutions seemed for several years to be an example. The cause was considered
to be the search space, whose cost function $E$ has a very irregular structure with isolated minima [Bernasconi 1987]. However, more recent approaches show that meta-heuristics can find optimal sequences efficiently. Examples of metaheuristic methods applied to LABS are simulated annealing [Bernasconi 1987, Golay 1982], evolutionary search [de Groot et al. 1992, Militzer et al. 1998, Mühlenbein 1991, Reinholz 1993, Wang 1987], TABU search [Douté and van Hentenryck 2006, Halim et al. 2008, Hulianytskyi and Sokol 1993], memetic algorithms [Gallardo et al. 2007], local search [Beenker et al. 1985, Brglez et al. 2003], and local search hybridised with relaxation [Prestwich 2007].

In this paper we find several ways of improving branch-and-bound for LABS, show that the new algorithm has improved scalability, and establish new optimality results for skew-symmetric sequences. The method is described in Section 2 and results are presented in Section 3.

2 Improved branch-and-bound for LABS

First we describe Mertens’ branch-and-bound method [Mertens 1996], then introduce our improvements.

2.1 Mertens’ method

In order to minimize the minimum energy

$$E_{\min} = \min \left( \sum_{k=1}^{N-1} C_k^2 \right)$$

of a partial sequence $A$, the relaxation

$$E^*_{\min} = \sum_{k=1}^{N-1} \min(C_k^2)$$

can be used as a lower bound $E^*_{\min} \leq E_{\min}$. Because $E^*_{\min}$ is still expensive to calculate, Mertens’ method uses a cheap lower bound $E_b \leq E^*_{\min}$ based on an arbitrary completion of the current partial sequence. Define a product $s_is_j$ to be *computable* if $s_i$ and $s_j$ are both assigned in the current partial sequence $A$. Let $l_k$ be the sum of its computable products and $f_k$ the number of its uncomputable products. A lower bound $l_k$ for each $C_k$ is calculated by finding the energy of the completed sequence then subtracting $2f_k$, because negating an element cannot reduce $C_k$ by more than 2. On finding a sequence with energy $E$ the search can proceed with upper bound $E - 4$ because it is known that these sequences have energies differing by multiples of 4 [Militzer et al. 1998].

A refinement exploits the fact that the sum of an odd number of $\pm 1$ values has absolute value at least 1. $l_k$ is refined to $\max(l_k, b_k)$ where $b_k = (N - k) \mod 2$. Then

$$E_b = \sum_{k=1}^{N-1} l_k^2$$
is a lower bound for \( E_{\text{min}}^* \).

Symmetry occurs in LABS because the energy of a sequence is unaffected if the sequence is reversed, if all its values are negated, or if oddly numbered values are negated: combining these three operations in all possible ways gives 8 equivalent sequences. If we can avoid exploring more than 1 sequence from each class we might reduce search effort by a factor approaching 8, and Mertens’ method achieves this by fixing the values of several of the outermost variables. The branching heuristic is chosen to facilitate symmetry breaking: variables are assigned outermost first, that is \( s_1, s_N, s_2, s_{N-1}, s_3, s_{N-2}, \ldots \) (actually they are treated as pairs \( (s_1, s_N), (s_2, s_{N-1}), \ldots \)).

2.2 Avoiding the use of an arbitrary sequence

In Mertens’ method the value of \( l_k \) depends on the arbitrary completion of the sequence, and its greatest possible value occurs when arbitrarily completing the sequence transforms each uncomputable product to \(-1\) in which case \( l_k = \max(b_k, |l_k| - f_k) \). But we can always achieve this value by reasoning as follows. To the known sum \( t_k \) of computable products we add \( f_k \) uncomputable products. If \( t_k > 0 \) then the worst case is that each uncomputable product is \(-1\), while if \( t_k < 0 \) the worst case is that each uncomputable product is \(1\) (if \( t_k = 0 \) then \( l_k = b_k \)), so we can use \( l_k = \max(b_k, |l_k| - f_k) \). This idea was previously used in a hybrid local search algorithm [Prestwich 2007].

2.3 Exploiting cancellations

\( f_k \) is the number of uncomputable products, but we can ignore some of these products because they can be predicted to cancel each other out. Suppose we have two products \( s_p s_q \) and \( s_q s_r \) where \( s_p, s_r \) have been assigned different values, but \( s_q \) has not yet been assigned. Whichever value \( s_q \) takes the two products will have different values, so we can subtract 2 from \( f_k \) without knowing the value of \( s_q \) (of course we must remember not to repeat this subtraction later in the search when \( s_q \) is assigned a value). We shall refer to this as a cancellation.

Cancellations occur if we use the same branching heuristic as Mertens. Suppose we have just assigned \( s_i \) and we compute some \( l_k \), where \( i \leq \lfloor N/2 \rfloor, i + 2k \leq N \), \( s_{i+k} \) is unassigned, \( s_{i+2k} \) is assigned, and \( s_i \neq s_i+2k \). Then we have a cancellation between products \( s_i s_{i+k} \) and \( s_{i+k} s_{i+2k} \). Similarly if \( i \geq \lceil N/2 \rceil, i - 2k \leq 1 \), \( s_{i-k} \) is unassigned, \( s_{i-2k} \) is assigned, and \( s_i \neq s_i-2k \), then we have a cancellation between products \( s_i s_{i-k} \) and \( s_{i-k} s_{i-2k} \). If we instead ordered variables \( (s_1, s_2, s_3, \ldots) \) then no cancellations would occur, so Mertens’ branching heuristic turns out to be ideal for cancellations as well as for symmetry breaking.

2.4 Exploiting reinforcements

We can also increase the value of \( b_k \) in some cases. Again consider two uncomputable products \( s_p s_q \) and \( s_q s_r \) where \( s_p, s_r \) have been assigned values but \( s_q \) has not.
This time suppose $s_p, s_r$ take the same value so that no cancellation occurs. Then the two uncomputable products will sum to either 2 or $-2$. We shall refer to this as a reinforcement. Again if we use Mertens’ branching heuristic reinforcements will often occur, and we exploit them as follows. If all uncomputable products occur either in cancellation or reinforcement pairs then $t_k$ is an even number. Now suppose that $t_k$ is also an even number, and that $t_k \text{mod} 4 \neq f_k \text{mod} 4$. Then $t_k + f_k$ is even but must be of the form $4i + 2$ for some integer $i \in \mathbb{Z}$ so we can set $b_k = 2$.

2.5 Value ordering

Our final improvement does not tighten the relaxation but leads to faster convergence. Mertens’ method presumably assigned each variable first to 1 then to $-1$ or vice-versa, because this is standard practice and no special value ordering was mentioned in [Mertens 1996]. In experiments we found that almost any other value ordering, including a randomised ordering, led more quickly to lower-energy sequences. We found best results by basing the value ordering on a known large sequence with low energy, as follows.

For each variable we choose a fixed value that will always be tried first during search: we shall refer to the vector of these values as a template. The template is based on a low-energy sequence found by local search so this is a simple way of exploiting local search results in branch-and-bound. To construct a template, for odd $N$ we take the middle $N$ values from the sequence

$$1211211211222B222111111112224542$$

which has length 67, energy 241 and merit factor 9.31, while for even $N$ we take the middle $N$ values from the sequence

$$11111111411472321232514121122221212$$

which has length 68, energy 250 and merit factor 9.25. These two sequences were chosen because they have high merit factors and are longer than any sequences we need in this paper. They are shown in run-length notation in which each number indicates the number of consecutive elements with the same value. For example the sequence $(1, 1, -1, 1, -1, -1, -1, -1, -1, 1)$ would be written $21141$: whether the sequence begins with 1 or $-1$ is irrelevant because of symmetry. For runs of length greater than 9 upper-case letters are used: A=10, B=11 and so on.

The motivation behind this idea is that each correlation $C_k$ for the new sequence takes all its terms from $C_k$ in the larger sequence. While this does not guarantee optimality it should lead to a low initial energy. In experiments this does indeed occur, and using a template greatly speeds up convergence to optimality. For example the graph in Figure 1 shows the runs for $N = 39$ with and without the use of a template. The use of a template results in much earlier low-energy sequences. It also results in far fewer distinct energies, which might aid future parallelisation by reducing communication between processes: 307 energies without a template and 17 with. The effect on runtime is significant in many cases. For example with $N = 39$ the method

\[1\] http://www.comp.nus.edu.sg/~stevenha/viz/results_labs.html
2.6 Symmetry

We break almost all symmetries in a similar way to Mertens, but taking the template into account via a standard technique from Constraint Programming. If we were using a constraint model we could break all symmetries by adding 7 lex-leader constraints [Crawford et al. 1996] to ensure that any sequence is the lexicographically-least in its class. For example to exploit the symmetry that results from reversing a sequence and negating its values we would add a constraint

$$\langle s_1, s_2, s_3, \ldots \rangle \preceq_{\text{lex}} \langle -s_N, -s_{N-1}, -s_{N-2}, \ldots \rangle$$

However, it is known that symmetry breaking can have a deleterious effect on search if it conflicts with the search heuristics: that is, if the excluded solutions are those that would have been found earliest without symmetry breaking. To avoid this conflict we ensure that the template is the lexicographically-least among all possible variable assignments, by adjusting the lex-leaders. For example the above lex-leader becomes

$$\langle s'_1, s'_2, s'_3, \ldots \rangle \preceq_{\text{lex}} \langle -s'_N, -s'_{N-1}, -s'_{N-2}, \ldots \rangle$$

where

$$s'_i = \begin{cases} s_i & \text{if } t_i = 1 \\ -s_i & \text{if } t_i = 0 \end{cases}$$
and \( t_i \) is template value \( i (i = 1 \ldots N) \). To reduce runtime overhead we do not use the lex-leaders at every search tree node. Instead we check that they are satisfied only at even-numbered depths down to a depth of \( N/2 \), which is sufficient to break most symmetry with low overhead.

### 2.7 Skew-symmetry

We can adapt our method to find only skew-symmetric sequences, which is the most common sieve for restricting search to a useful subset of all sequences [Golay 1982]. The skew-symmetric sequences have odd length with \( N = 2n - 1 \) for some \( n \), and satisfy

\[
 s_{n+i} = (-1)^i s_{n-i} \quad (i = 1 \ldots n-1)
\]

This roughly halves the number of independent variables in the problem, which greatly reduces the search space. Such sequences often have good merit factors because \( C_k = 0 \) for all odd \( k \). (Note that skew-symmetry is a property of a single sequence, and should not be confused with the 8-fold symmetry between sequences described above.)

Optimal skew-symmetric sequences have been enumerated using branch-and-bound for \( N \leq 71 \) by de Groot et al. [1992] and for \( N \leq 73 \) by Reinholz [1993], and good solutions for larger \( N \) have been found using metaheuristics [Beenker et al. 1985, de Groot et al. 1992, Golay and Harris 1990, Militzer et al. 1998, Mühlenbein 1991, Prestwich 2007, Reinholz 1993, Wang 1987].

To adapt our branch-and-bound method to skew-symmetric sequences we need three modifications. Firstly we ensure that no assignment violates skew-symmetry. Secondly on finding a sequence with energy \( E \) we can use a new upper bound \( E - 8 \). Thirdly we need a longer template that is also skew-symmetric, and we use a known sequence with \( N = 119 \):

\[
11331111311332321211561311512
\]

(only the first 60 values are represented here as the rest can be deduced by skew-symmetry) which has energy 835 and merit factor 8.48.

### 3 Results and conclusions

Mertens tested the scalability of the branch-and-bound method by counting the number of recursive calls needed to find an optimal sequence and prove it optimal, using results for \( N = 15 \ldots 44 \) then curve-fitting (H. Bauke, personal communication) and found that it needed \( O(1.85^N) \) calls. Performing the same experiment we find improved scalability of \( O(1.74^N) \) calls, or \( O(1.80^N) \) seconds of wall clock time, using a C implementation of our method executed on a 2.8 GHz Pentium 4 with 512 MB RAM. We hope to use this method to find new optimal sequences in the future, but to do this we need a parallel implementation: since publishing [Mertens 1996] Mertens and Bauke have found provably optimal sequences up to \( N = 60 \) using a cluster of 160 processors. By extrapolation we do not expect a speedup of 160 to occur until \( N \approx 83 \) so for the present parallelism trumps our improvements. But our new method
should give good results when implemented on a cluster and we hope to try this in future work.

However, we can use our method to find new results for skew-symmetric sequences, which to the best of our knowledge have not been attacked using highly parallel hardware. In experiments we found no new skew-symmetric sequences, but confirmed the optimality of several published sequences previously found by metaheuristics. Table 1 shows the merit factors and execution times. Results for $N \leq 71$ can be found in [de Groot et al. 1992], and because [Reinholz 1993] may not be easy to obtain we mention that the optimal merit factor for skew-symmetric sequences of length $N = 73$ is 7.66. The merit factor 8.25 for $N = 75$ is optimal though [Beenker et al. 1985] incorrectly gives it as 9.25.

There might be further possible improvements to the LABS branch-and-bound algorithm. If we expand the energy expression we obtain a quartic polynomial which in principle allows more cancellations. Consider terms $s_a s_b s_c s_d$ and $s_a s_b s_c s_e$. If $s_a, s_b, s_d, s_e$ are assigned and if $s_a s_b s_d \neq s_a s_b s_e$, then the two terms cancel out whatever the value of $s_c$. Furthermore, if only $s_a, s_d, s_e$ are assigned and $s_a s_d \neq s_a s_e$, the terms cancel out whatever the values of $s_b, s_e$. And if only $s_d, s_e$ are assigned and $s_d \neq s_e$ then the two terms cancel out whatever the values of $s_a, s_b, s_c$. The difficulty lies in exploiting these additional cancellations efficiently, which is an interesting possibility for future research.

Table 1 New optimality results for skew-symmetric sequences

| $N$ | $F$ | seconds |
|-----|-----|---------|
| 75  | 8.25| 3,655   |
| 77  | 8.28| 9,140   |
| 79  | 7.67| 17,889  |
| 81  | 8.20| 28,321  |
| 85  | 8.17| 74,994  |
| 87  | 8.39| 143,147 |
| 89  | 8.18| 285,326 |

References

Beenker et al. 1985. Beenker, G., Claasen, T., & Hermens, P. (1985). Binary Sequences With a Maximally Flat Amplitude Spectrum. *Philips J. of Research* 40, 289–304.

Bernasconi 1987. Bernasconi, J. (1987). Low Autocorrelation Binary Sequences: Statistical Mechanics and Configuration Space Analysis. *J. Physique* 48, 559–567.

Brglez et al. 2003. Brglez, F., Li, X. Y., Stallow, M. F., & Militzer, B. (2003). Reliable Cost Prediction for Finding Optimal Solutions to LABS Problem: Evolutionary and Alternative Algorithms. *Fifth International Workshop on Frontier in Evolutionary Algorithms*, Cary, NC, USA.

Crawford et al. 1996. Crawford, J., Ginsberg, M. L., Luks, E., & Roy, A. (1996). Symmetry-Breaking Predicates for Search Problems. *Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, pp. 148–159.

de Groot et al. 1992. de Groot, C., Würz, D., & Hoffmann, K. H. (1992). Low Autocorrelation Binary Sequences: Exact Enumeration and Optimization by Evolutionary Strategies. *Optimization* 23, 369–384.
Dotú and van Hentenryck 2006. Dotú, I., van Hentenryck, P. (2006). A Note on Low Autocorrelation Binary Sequences. *Principles and Practice of Constraint Programming, Lecture Notes in Computer Science* 4204:685–689.

Gallardo et al. 2007. Gallardo, J. E., Cotta, C., & Fernandez, A. J. (2007). A Memetic Algorithm for the Low Autocorrelation Binary Sequence Problem. *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation* pp. 1226–1233.

Gent and Walsh 1999. Gent, I. P., & Walsh, T. (1999). CSPLib: A Benchmark Library for Constraints. Available on [http://www.csplib.org/](http://www.csplib.org/).

Halim et al. 2008. Halim, S., Yap, R., & Halim, F. (2008) Engineering Stochastic Local Search for the Low Autocorrelation Binary Sequence Problem. *Principles and Practice of Constraint Programming, Lecture Notes in Computer Science* 5202:640–645.

Hulianytskyi and Sokol 1993. Hulianytskyi, L., & Sokol, V . (1993). Effective Energy Recomputation for Low Autocorrelation Binary Sequence Problem. *Information Theories and Applications* 18(4):344–357.

Kratica 2012. Kratica, J. (2012). A Mixed Integer Quadratic Programming Model for the Low Autocorrelation Binary Sequence Problem. *Serdica Journal of Computing* 6:385–400.

Mertens 1996. Mertens, S. (1996). Exhaustive Search for Low-Autocorrelation Binary Sequences. *J. Phys. A: Math. Gen.* 29, L473–L481.

Mertens and Bessenrodt 1998. Mertens, S., & Bessenrodt, C. (1998). On the Ground States of the Bernasconi Model. *J. Phys. A: Math. Gen.* 31, 3731–3749.

Militzer et al. 1998. Militzer, B., Zamperelli, M., & Beule, D. (1998). Evolutionary Search for Low Autocorrelated Binary Sequences. *IEEE Trans. Evol. Comp.* 2, 34–39.

Mühlenbein 1991. Mühlenbein, H. (1991). Asynchronous Parallel Search by the Parallel Genetic Algorithm. *Third IEEE Symposium on Parallel and Distributed Processing*, IEEE Computer Society Press, 526–33.

Prestwich 2007. Prestwich, S. D. (2007). Exploiting Relaxation in Local Search for LABS. *Annals of Operations Research* 156(1):129–141.

Reinholz 1993. Reinholz, A. (1993). Ein Paralleler Genetischer Algorithmus zur Optimierung der Binären Autokorrelations-Funktion. Diplom thesis, Universität Bonn.

Wang 1987. Wang, Q. (1987). Optimization by Simulating Molecular Evolution. *Biol. Cybern.* 57:95–101.