CYBERNETICS AND PHYSICS, VOL. 15, NO. 1, 2026, 112–117

DISCRETE OPTIMIZATION FOR BINARY PHASE-MANIPULATED SIGNALS: CONSTRUCTING LONG ±1 SEQUENCES WITH LOW APERIODIC AUTOCORRELATION

Bo Zhang
Shenzhen MSU-BIT University
Shenzhen, China
e-mail: zhangbo@smbu.edu.cn

Boris F. Melnikov
Shenzhen MSU-BIT University
Shenzhen, China
Corresponding author; e-mail: bormel@mail.ru

Article history:
Received 27.05.2026, Accepted 19.06.2026

Abstract

Binary phase-manipulated probe signals (BPM, or BPSK) are widely used for echo detection and pulse compression. Reliable detection of the full return time of a long probe requires a sharp matched-filter peak at the correct delay and near-zero responses at other delays. This requirement leads to the synthesis of long binary sequences whose aperiodic autocorrelation has a dominant zero-shift peak and very small sidelobes. We formulate the task as a discrete optimization problem over length N sequences taking values in the set $\{-1, +1\}$ and propose a practical construction strategy that combines (i) exhaustive enumeration of near-optimal short blocks, (ii) symmetry augmentation (reversal and sign inversion), and (iii) greedy/beam splicing to build long sequences. The method is simple to implement, naturally parallelizable, and improves the normalized sidelobe-energy objective (phi), ISL, and PSL over an optimistic random baseline (best-of-200 trials), while a genetic-algorithm baseline can reach lower ISL at a substantially higher number of objective evaluations.

Key words

binary phase coding; aperiodic autocorrelation; LABS; discrete optimization; beam search; genetic algorithms

1 Introduction

Binary phase-coded probing is a classical technique in radar/sonar and spread-spectrum communications: a known binary sequence modulates a carrier phase, and the receiver applies a matched filter (correlation) to detect the embedded code and estimate the delay [Levanon and Mozeson, 2004; Skolnik, 2008]. In the presence of noise and interference, detection quality is largely controlled by the sidelobes of the aperiodic autocorrelation function (AACF). Very low sidelobes reduce false detections and prevent strong echoes/clutter from masking weak targets [Levanon and Mozeson, 2004; Lin et al., 2019]. Related correlation-noise minimization for BFM/BPSK probing using genetic algorithms and unmatched filters has also been studied [Lozovoy et al., 2000].

From the physical point of view, the optimization criterion is directly related to the energy distribution of the matched-filter output. The main peak at zero shift corresponds to the physically correct delay of the returned probe, whereas nonzero-shift autocorrelation values correspond to undesired responses at wrong delays. Therefore, a sequence with small aperiodic sidelobes reduces ambiguity in echo detection and improves the separation of weak reflected signals from correlation noise and clutter. This setting is relevant not only for radar and sonar pulse compression, but also for acoustic and ultrasound measurement systems, where the calibration and interpretation of received signals depend on accurate delay estimation. Recent CAP studies of ultrasound transceiver-array calibration and parameter estimation under unknown-but-bounded noise illustrate related measurement settings in which signal design, noise suppression, and reconstruction accuracy are central issues [Granichin et al., 2025; Len et al., 2025].

For short lengths, Barker sequences are famous examples with exceptionally low sidelobes, but they exist only for very limited lengths (up to 13 in the binary case) [Jedwab, 2005; Ukil, 2010]. For long lengths, the synthesis problem is computationally hard and is closely related to the Low Autocorrelation Binary Sequence (LABS) problem studied in optimization and statistical physics [Mertens, 1996; Prestwich, 2013; Packebusch and Mertens, 2016]. Accordingly,

both complete-search strategies (branch-and-bound) and stochastic/metaheuristic methods (evolutionary search, local search, GPU acceleration) have been explored [Halim et al., 2008; Prestwich, 2013; Bošković and Brest, 2024]. An anytime multiheuristic viewpoint combining truncated branch-and-bound and evolutionary search has also been advocated in discrete optimization [Melnikov, 2005; Melnikov, 2006; Melnikov et al., 2018].

This paper focuses on a pragmatic discrete-optimization viewpoint motivated by a simple physical narrative: a very long binary probe is transmitted; the reflected probe is observed as a noisy fragment in a long received stream; the goal is to determine when the probe has *fully returned*. Operationally, one slides a copy of the transmitted code across the received stream and computes correlation; a single sharp maximum indicates the correct delay. Hence, we seek binary codes that are maximally "unlike" their own shifts: for all nonzero shifts the correlation should be near zero.

Our contribution is a construction strategy tailored for fast implementation: we first enumerate near-optimal short blocks, then assemble long sequences by greedy/beam splicing while directly optimizing standard sidelobe metrics.

2 Problem formulation

Let $s = (s_1, \dots, s_N)$ be a binary sequence with $s_i \in \{-1, +1\}$. The *aperiodic autocorrelation* at shift k is

$$C_s(k) = \sum_{i=1}^{N-k} s_i\, s_{i+k}, \qquad k = 0, 1, \dots, N-1, \quad (1)$$

with $C_s(0) = N$. Two widely used sidelobe criteria are:

$$\mathrm{PSL}(s) = \max_{1 \le k \le N-1} |C_s(k)|, \quad (2)$$

$$\mathrm{ISL}(s) = \sum_{k=1}^{N-1} C_s(k)^2. \quad (3)$$

A normalized objective is

$$\phi(s) = \frac{1}{N}\sqrt{\sum_{k=1}^{N-1} C_s(k)^2} = \frac{1}{N}\sqrt{\mathrm{ISL}(s)}. \quad (4)$$

The value ϕ has a simple practical interpretation. Since $C_s(k)$ for $k \neq 0$ is the matched-filter response at an incorrect delay, ISL is the total sidelobe energy outside the main peak. Thus ϕ is the root-mean-square sidelobe level normalized by the sequence length. Smaller ϕ means that, on average, wrong-delay responses are weaker relative to the main response $C_s(0) = N$. In physical detection problems this corresponds to a lower correlation-noise background and a smaller probability that a sidelobe will be confused with the true echo.

Minimizing ϕ is equivalent to minimizing ISL. Another classical quality measure is Golay's *merit factor* $F(s) = N^2/(2\,\mathrm{ISL}(s))$ [Golay, 1982; Jedwab, 2005].

Therefore, the discrete optimization problem studied here is as follows:

$$\min_{s \in \{\pm 1\}^N} \phi(s) \quad (5)$$

(or equivalently minimize ISL / PSL). The search space size is 2^N (e.g., 2^{128}), which excludes exhaustive enumeration for long N.

3 Methods

3.1 Enumerating near-optimal short blocks

For moderate block length m (e.g., $m = 16$–20), exhaustive enumeration is feasible ($2^{16} = 65536$). We compute ϕ (or PSL/ISL) for every block and retain the best K blocks. To expand the candidate pool, we apply simple symmetries: *reversal* and *sign inversion* (and their combination). Although sign inversion leaves autocorrelation unchanged, it can improve boundary matching in a splicing construction.

3.2 Greedy splicing with overlap

Let the current partial sequence be x and a candidate block be b. We use an overlap length r (default $r = \lfloor m/4 \rfloor$) and measure boundary compatibility by

$$\mathrm{sim}(x, b) = \langle x_{|x|-r+1:|x|},\, b_{1:r} \rangle = \sum_{i=1}^{r} x_{|x|-r+i}\, b_i.$$

To accelerate search, one may preselect the top P blocks with the largest $\mathrm{sim}(x, b)$ and then score only these candidates by the exact objective (ISL or ϕ) after appending b (dropping the first r symbols of b). In our experiments, the candidate pool is small (after augmentation), so we evaluate all candidates exactly.

3.3 Beam search as practical branch-and-bound

Greedy splicing can be myopic. We therefore keep a beam (bounded frontier) of size B. At each step, every beam element is expanded by candidate blocks, and the expanded partial sequences are ranked by ISL (equivalently by ϕ, since $\phi = \sqrt{\mathrm{ISL}}/N$). We keep the best B partial sequences and repeat until length N is reached. This is a practical branch-and-bound style heuristic with an explicit memory budget, closely related in spirit to truncated/anytime branch-and-bound with heuristic guidance [Melnikov, 2006; Melnikov, 2005].

3.4 Baseline: genetic algorithm

For comparison, we also implement a standard genetic algorithm (GA) on $\{\pm 1\}^N$: tournament selection, single-point crossover, bit-flip mutation, and elitism. GAs have been repeatedly used for BFM/BPSK correlation-noise reduction settings [Lozovoy et al., 2000] and also in sequence synthesis [Bošković and Brest, 2024].

Table 1. Comparison of methods on different lengths N. Lower ϕ, ISL and PSL are better; larger merit factor F is better. (Splice: beam splicing with $B = 30$; Random: best of 200 trials; GA: population $P = 200$, generations $G = 300$.)

N	method	ϕ	PSL	ISL	F
64	Splice ($B = 30$)	0.458	9	860	2.381
64	Random (best/200)	0.502	10	1032	1.984
64	GA ($P = 200$, $G = 300$)	0.366	9	548	3.737
100	Splice ($B = 30$)	0.471	15	2222	2.250
100	Random (best/200)	0.534	18	2854	1.752
100	GA ($P = 200$, $G = 300$)	0.351	11	1234	4.052
128	Splice ($B = 30$)	0.459	17	3456	2.370
128	Random (best/200)	0.518	19	4404	1.860
128	GA ($P = 200$, $G = 300$)	0.351	11	2024	4.047

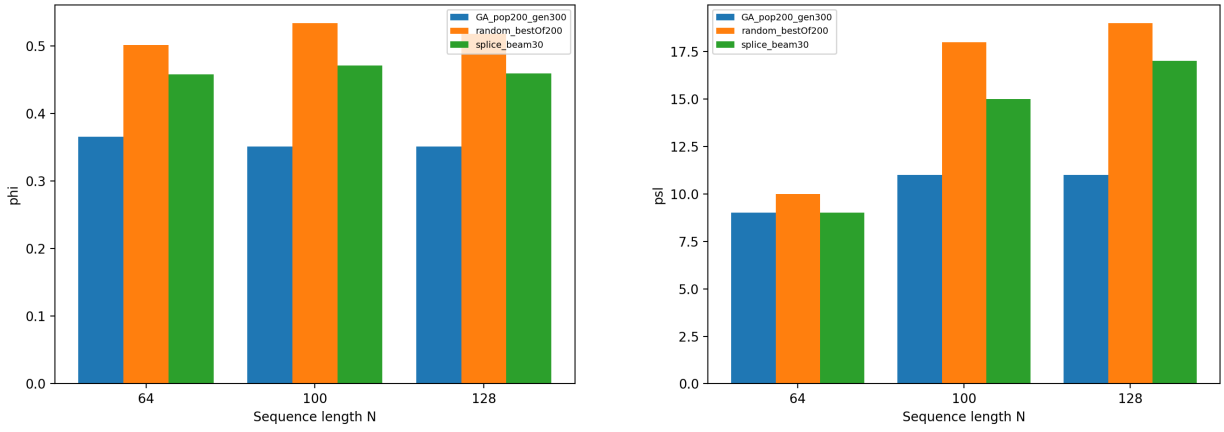

Figure 1. Performance comparison across methods and lengths: (left) normalized sidelobe energy objective ϕ; (right) peak sidelobe level (PSL).

4 Experiments

All experiments are implemented in Python (NumPy) and are fully reproducible via the scripts released with this paper. We evaluate the proposed splicing/beam method against (i) random sequences and (ii) GA baselines for lengths $N \in \{64, 100, 128\}$. For each method, we report ϕ, PSL, and ISL; we also report the merit factor. The three lengths were selected to probe different practical and computational regimes. The case $N = 64$ represents a relatively short probing code and enables a quick comparison of the construction strategies. The case $N = 100$ is a non-power-of-two intermediate length, which checks that the procedure is not tuned only to dyadic lengths. The case $N = 128$ is a longer test case for which exhaustive enumeration of all binary sequences is infeasible. Together, these lengths demonstrate the scaling tendency and the complexity-quality tradeoff of the method; they are not intended as evidence of global optimality for all long sequences.

4.1 How to reproduce

A self-contained reproduction instruction is provided in Appendix B.

4.2 Results

Table 1 summarizes the results for $N \in \{64, 100, 128\}$. The beam-splice construction consistently improves over an optimistic random baseline (best of 200 random trials) for all tested lengths. For instance, at $N = 128$ it reduces ϕ from 0.518458 to 0.459279 (about 11.4%) and ISL from 4404 to 3456 (about 21.5%), while also decreasing PSL from 19 to 17.

The GA baseline achieves the best ISL (and thus the best ϕ) in these runs (e.g., $\phi \approx 0.351$ for $N = 100$ and $N = 128$), but it requires substantially more objective evaluations. With population size 200 and 300 generations, GA evaluates on the order of 6×10^4 candidates per N. In contrast, the beam-splice method operates on a small augmented block pool (here $m = 16$, $K = 10$ with reversal/sign symmetries, yielding up to $4K$ candidates) and provides a fast, deterministic construction that is attractive when rapid prototyping and interpretability are desired.

4.3 Computational cost and quality tradeoff

The reported runs show a clear quality-complexity tradeoff. GA obtains better final ISL, PSL, and ϕ values, so it is the stronger optimizer for these three lengths. The role of `Splice` is different: it is preferable when the goal is a fast, deterministic, transparent, and easily parallelizable construction with fewer objective evaluations. This distinction is important in expensive-evaluation settings, and recent CAP work on rejection-based genetic algorithms provides related context for evolutionary search with controlled candidate acceptance [Deeb et al., 2025].

The one-time block-enumeration stage requires 2^m evaluations for block length m; for $m = 16$ this is 65536 short-block evaluations and can be reused for all tested N. Let $A \leq 4K$ be the augmented candidate-pool size after reversal and sign inversion. With $m = 16$, $K = 10$, $A = 40$, beam width $B = 30$, and overlap $r = 4$, the number of beam-expansion objective evaluations is approximately

$$A + BA \left\lceil \frac{N-m}{m-r} \right\rceil,$$

when all augmented blocks are evaluated at each splicing step. This gives 4840, 8440, and 12040 evaluations for $N = 64, 100, 128$, respectively. By comparison, the GA baseline with population $P = 200$ and $G = 300$ performs about $PG = 60000$ objective evaluations for each length. The formula also shows the parameter scaling: increasing B or K increases the expansion budget linearly, increasing m increases the one-time enumeration cost exponentially, and increasing the overlap r increases the number of splicing steps because each appended block contributes only $m - r$ new symbols.

As a verification run, the scripts were executed in a single Python process on a local arm64 macOS system (Darwin 25.3.0, 8 logical CPU cores) using Python 3.12.13 and NumPy 2.3.5. The measured wall-clock times for `Splice` were approximately 0.16, 0.39, and 0.68 seconds for $N = 64, 100, 128$, respectively; the corresponding GA times were approximately 3.35, 4.73, and 5.92 seconds. These times are implementation- and hardware-dependent, whereas the objective-evaluation counts above are hardware-independent.

For reproducibility, the optimization output is extremely compact (a single ±1 vector). Therefore, for every reported computation variant, we list the resulting vector explicitly in Appendix A (one vector per method and per N). In our tests, the proposed `Splice` construction is not the best method in terms of the final ISL or PSL values. Its role is instead to provide a fast, transparent, and easily reproducible baseline construction. This distinction is important: GA is stronger as an optimizer, while `Splice` is more convenient as a deterministic constructive procedure with a smaller evaluation budget.

5 Discussion

The proposed pipeline trades algebraic optimality for implementability and flexibility. It is naturally parallel: block enumeration, beam expansions, and GA fitness evaluation can be distributed across cores/GPUs. To the best of our knowledge, the overlap-based beam-splicing procedure described here has not been presented as a standard construction in the LABS literature. We therefore treat it as a new constructive heuristic baseline rather than as a replacement for stronger stochastic optimizers.

The physical contribution of this work lies in the computational design of binary phase-manipulated probing signals for measurement and detection systems. In such systems, autocorrelation sidelobes appear as wrong-delay matched-filter responses, so reducing sidelobe energy directly improves delay estimation, echo localization, and the separation of weak physical returns from clutter and correlation noise. Thus the paper contributes to computational signal design for physical probing and matched-filter-based detection.

Future work includes:

(i) seeding with algebraic constructions (e.g., Legendre-related seeds),
(ii) tighter bounds for complete search, and
(iii) alternative correlation measures that handle ties and quantization effects.

A broader graph viewpoint can also be useful: correlations are inner products of shifted vectors, and in related settings, spectral characteristics of Laplacian matrices govern the stability and convergence of decentralized algorithms. A recent example of such spectral analysis published in this journal is [Erofeeva and Parsegov, 2024]. Similarly, when such graph models are treated as random objects within specified classes, statistical study of their invariants can support heuristic characterization; see [Melnikov and Liu, 2025].

6 Conclusion

We formulated the synthesis of long BPM/BFM probing codes as a discrete optimization problem over $\{\pm 1\}^N$ and provided a practical construction method based on enumerated short blocks and greedy/beam splicing. The approach supports rapid experimentation and provides a transparent baseline for further refinement of low-autocorrelation binary sequence construction methods.

A Result vectors for each computation variant

In this appendix, we encode each output vector as a binary string of length N with the mapping:

$$-1 \mapsto \texttt{0} \quad \text{and} \quad +1 \mapsto \texttt{1}.$$

$N = 64$

splice_beam30:

```
00110000010101100101000000110111
01001011100001000100001000011110
```

random_bestOf200:

```
10011001000001001101010001101001
11001111011010110100000110000001
```

GA_pop200_gen300:

```
11011101101010100001011011010011
11110001110001110111000100010011
```

$N = 100$

splice_beam30:

```
00110000010101100101000000111110
11011101101000011000010000111110
10000100010001110100101100111001 1010
```

random_bestOf200:

```
11010000010010101111110000100110
00101100101010100010010111101110
01101111001101010001110001100000 1100
```

GA_pop200_gen300:

```
00101000101111100101010111111001
10010101111001110110010001101101
00100001110001101111110011110000 0100
```

$N = 128$

splice_beam30:

```
00110000010101100101000000111110
11011101101000011000010000111110
10000100010001110100101100111001
10101111010110010101111100110110
```

random_bestOf200:

```
00110100101010010100110100111110
10101010110001110110000111100111
00110001101001011000001000101111
10011011001000000101010010010000
```

GA_pop200_gen300:

```
00101110111001010111001110001010
01000101100011000111010010100101
11101001011001111111001011100010
11101111111110100101100100000111
```

B Instruction for reproduction

This appendix contains the instructions only for reproducing Table 1 and Fig. 1. The source code used for reproducing the experiments is included in the submitted source archive. Default settings used in this paper are $m = 16$, $K = 10$, $B = 30$, overlap $r = 4$, candidate preselection disabled (`--preselect 0`), random trials $= 200$, GA population $P = 200$, and generations $G = 300$.

The typical workflow is:

(i) enumerate near-optimal short blocks and save them to `results/blocks_m16_top10_isl.npz`;

(ii) construct long sequences by beam splicing for $N \in \{64, 100, 128\}$ and save the resulting vectors in the `results` directory;

(iii) run the random and GA baselines and write `results/summary.csv` (and `results/summary_table.tex`);

(iv) and generate the plots `fig/summary_phi.png` and `fig/summary_psl.png`.

A minimal command-line example is:

```
python3 code/enumerate_blocks.py --m 16 --K 10
--metric isl --augment --out results
python3 code/construct_splice.py --blocks
results/blocks_m16_top10_isl.npz --N 64 --beam
30 --preselect 0 --out results
python3 code/construct_splice.py --blocks
results/blocks_m16_top10_isl.npz --N 100 --beam
30 --preselect 0 --out results
python3 code/construct_splice.py --blocks
results/blocks_m16_top10_isl.npz --N 128 --beam
30 --preselect 0 --out results
python3 code/run_experiments.py --blocks
results/blocks_m16_top10_isl.npz --Ns 64 100 128
--beam 30 --preselect 0 \
   --random_trials 200 --ga_pop 200 --ga_gen
300 --seed 0 --out results
python3 code/plot_results.py --csv
results/summary.csv --out fig
```

References

Bošković, B. and Brest, J. (2024). Two-phase optimization of binary sequences with low peak sidelobe level value. *Expert Systems with Applications*, **251**, pp. 124032.

Deeb, A., Khokhlovskiy, V., and Shkodyrev, V. (2025). Enhancing evolutionary controllers with a rejection-based genetic algorithm. *Cybernetics and Physics*, **14** (4), pp. 325–333.

Erofeeva, V. and Parsegov, S. (2024). Properties of the laplacian spectra of certain basic and hierarchical graphs. *Cybernetics and Physics*, **13** (1), pp. 12–19.

Golay, M. J. E. (1982). The merit factor of long low autocorrelation binary sequences. *IEEE Transactions on Information Theory*, **28** (3), pp. 543–549.

Granichin, O., Shcherbakov, P., Granichina, O., Trofimov, S., and Yuchi, M. (2025). Towards calibration of circular ultrasound transceiver arrays. *Cybernetics and Physics*, **14** (4), pp. 334–341.

Halim, S., Yap, R. H. C., and Halim, F. (2008). Engineering stochastic local search for the low autocorrelation binary sequence problem. In *Principles and Practice of Constraint Programming (CP 2008)*, vol. 5202 of *Lecture Notes in Computer Science*, pp. 640–645. Springer.

Jedwab, J. (2005). A survey of the merit factor problem for binary sequences. In *Sequences and Their Applications – SETA 2004*, vol. 3486 of *Lecture Notes in Computer Science*, pp. 30–55. Springer.

Len, I., Erofeeva, V., Granichin, O., and Smetanina, V. (2025). Parameter estimation using compressed sensing under unknown-but-bounded noise. *Cybernetics and Physics*, **14** (4), pp. 342–349.

Levanon, N. and Mozeson, E. (2004). *Radar Signals*. Wiley-IEEE Press.

Lin, R., Soltanalian, M., Tang, B., and Li, J. (2019). Efficient design of binary sequences with low autocorrelation sidelobes. *IEEE Transactions on Signal Processing*, **67** (24), pp. 6397–6410.

Lozovoy, A. V., Melnikov, B. F., and Radionov, A. N. (2000). The use of genetic algorithms and specific unmatched n-filters for BPSK signals correlation

noise minimizing. In *Proceedings of the 3rd International Scientific Conference and Exhibition "Digital Signal Processing and its Application" (DSPA-2000)*, Moscow, Russia, Insvyaz'izdat, pp. 144–149.

Melnikov, B. (2005). Discrete optimization problems – some new heuristic approaches. In *Proceedings of the Eighth International Conference on High-Performance Computing in Asia-Pacific Region (HPCAsia 2005)*, Beijing, China, IEEE, pp. 73–80.

Melnikov, B. and Liu, B. (2025). Special generation of random graphs and statistical study of some of their invariants. *Mathematics*, **13** (12), pp. 1904.

Melnikov, B. F. (2006). Multiheuristic approach to discrete optimization problems. *Cybernetics and Systems Analysis*, **42** (3), pp. 335–341.

Melnikov, B. F., Melnikova, E. A., Pivneva, S. V., Dudnikov, V. A., and Davydova, E. V. (2018). Geometric and game approaches for some discrete optimization problems. In *Data Science. Information Technology and Nanotechnology (DS-ITNT 2018): Proceedings of the International Conference Information Technology and Nanotechnology, Session Data Science*, vol. 2212 of *CEUR Workshop Proceedings*, pp. 312–321.

Mertens, S. (1996). Exhaustive search for low-autocorrelation binary sequences. *Journal of Physics A: Mathematical and General*, **29**, pp. L473–L481.

Packebusch, T. and Mertens, S. (2016). Low autocorrelation binary sequences. *Journal of Physics A: Mathematical and Theoretical*, **49** (16), pp. 165001.

Prestwich, S. D. (2013). Improved branch-and-bound for low autocorrelation binary sequences. *arXiv*. arXiv:1305.6187.

Skolnik, M. I. (2008). *Radar Handbook*. McGraw-Hill Education, 3 edition.

Ukil, A. (2010). Low autocorrelation binary sequences: Number theory-based analysis for minimum energy level, barker codes. *Digital Signal Processing*, **20** (2), pp. 483–495.